The Focused Inverse Method for Linear Logic

Kaustuv Chaudhuri

CMU-CS-06-162

December 4, 2006

Department of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **04 DEC 2006**	2. REPORT TYPE	3. DATES COVERED **00-00-2006 to 00-00-2006**
4. TITLE AND SUBTITLE **The Focused Inverse Method for Linear Logic**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Computer Science,Carnegie Mellon University,Pittsburgh,PA,15213**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
http://reports-archive.adm.cs.cmu.edu/anon/2006/CMU-CS-06-162.pdf

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**222**	

This work has been supported by the Office of Naval Research (ONR) under grant MURI N00014-04-1-0724 and by the National Science Foundation (NSF) under grant CCR-0306313. The views expressed in this document are those of the author, and do not necessarily reflect the policies or opinions of the ONR, the NSF, the United States government, Carnegie Mellon University, or any other entity.

Abstract

Linear logic presents a unified framework for describing and reasoning about stateful systems. Because of its view of hypotheses as resources, it supports such phenomena as concurrency, external and internal choice, and state transitions that are common in such domains as protocol verification, concurrent computation, process calculi and games. It accomplishes this unifying view by providing logical connectives whose behaviour is closely tied to the precise collection of resources. The interaction of the rules for multiplicative, additive and exponential connectives gives rise to a wide and expressive array of behaviours. This expressivity comes with a price: even simple fragments of the logic are highly complex or undecidable.

Various approaches have been taken to produce automated reasoning systems for fragments of linear logic. This thesis addresses the need for automated reasoning for the *complete* set of connectives for first-order intuitionistic linear logic ($\otimes, \mathbf{1}, \multimap, \&, \top, \oplus, \mathbf{0}, !, \forall, \exists$), which removes the need for any idiomatic constructions in smaller fragments and instead allows direct logical expression. The particular theorem proving technique used is a novel combination of a variant of Maslov's inverse method using Andreoli's focused derivations in the sequent calculus as the underlying framework.

The goal of this thesis is to establish the focused inverse method as the premier means of automated reasoning in linear logic. To this end, the technical claims are substantiated with an implementation of a competitive first-order theorem prover for linear logic – as of this writing, the only one of its kind.

Contents

Acknowledgements

I am very grateful to the people who have provided help and guidance to me in the course of this work. I am immensely grateful to my thesis advisor, Frank Pfenning, for setting the technical stage and motivation for much of this work, and for the years of advice, support and encouragement he has patiently provided to me. I also thank and commend the Department of Computer Science at Carnegie Mellon, not only for maintaining and encouraging a high quality of work from their graduate students, but also for providing an extraordinarily supportive enviroment where the students can truly pursue their work free of any worries.

I would also like to thank the other members of my thesis committee, Tanel Tammet, Jeremy Avigad, and Stephen Brookes, for their review and helpful comments on this thesis. I thank Dale Miller and Chuck Liang for some useful discussions and feedback on the penultimate draft of the thesis.

Special thanks and well wishes to Brigitte Pientka, Kevin Watkins, Uluc Saranli, Jason Reed, Martin Zinkevich, Noam Zeilberger, Deepak Garg, Sean McLaughlin, Daniel Blandford, and my colleagues and fellow students at CMU, for discussions, collaboration and camaraderie. Also, a cheerful nod to Pittsburgh; despite its many oddities, I have become rather fond of this unassuming city.

Chapter 1

Introduction

1.1 Thesis

The combination of the inverse method and focused derivations gives a viable and efficient means of automated proof search in first-order linear logic.

1.2 Motivation

This thesis is devoted to building a general theorem prover for linear logic [42] using the inverse method [73, 116]. Linear logic differs from natural logic by disallowing the structural operations of weakening and contraction, either as rules of inference or as admissible structural theorems. The number of occurrences of hypotheses thus plays a critical role in the proof theory, allowing encodings of precise counting semantics. A linear hypothesis must have *exactly one use* in a proof, which supports a view of a linear assumptions as *resources*, and proofs as *consumers* of such resources. This thesis is primarily concerned with intuitionistic linear logic, which maintains a separation between plural resources and singular conclusion, breaking the symmetry present in classical linear logic, but making finer distinctions between some connectives (for instance, & and $\oplus$ are not definable in terms of each other) and disallowing some classical connectives such as multiplicative disjunction ($\parr$). Of particular import is the elevation of linear implication ($\multimap$) to the status of a logical connective independent of other connectives.

The richness of intuitionistic linear logic makes the theorem proving problem harder than for classical linear logic; however, there is no fundamental reason why the approaches in this thesis cannot apply to the classical case. Additionally, the theory of proofs for intuitionistic linear logic, and particularly checking these proofs efficiently, is very well understood and forms the core of *logical frameworks* such as CLF [117]. Needless to say, producing checkable proofs increases confidence in the theorem prover. The different set of connectives in intuitionistic linear logic does not make it any less expressive than classical linear logic [27].

The linear view of assumptions as resources has led to various applications in reasoning about state; for example, planning (finite and infinite state) [71], protocol verification [14, 17, 16], concurrent computation [5], process calculi [77], Petri-nets [20], security protocols [25] and games [62, 6]. It is remarkable that linear logic serves as a uniform language for such diverse systems. Automated deduction for linear logic therefore has wide-ranging appeal. Besides allowing a direct analysis of the logic of systems, linear logic offers a foundational account of such phenomena as internal/external choice or true concurrency that occur naturally in stateful concurrent systems. Such phenomena have to be engineered into other descriptive frameworks, such as CCS processes, without clear logical motivations, and often requires operations such as predicate abstraction [47] in order to obtain tractable model checking problems. These frameworks must therefore be understood as indirect reasoning systems; in fact, they are not in competition with automated theorem proving, but rather present a wealth of alternative approaches for situations where theorem provers fail to perform efficiently. Logical reasoning, however, should come first.

The novelty of automated theorem proving in linear logic lies in handling resources efficiently – the *resource management problem*. The source of this problem is the lack of structural weakening and contraction, which makes even propositional linear logic undecidable [68]. Resource management for backward linear logic programming has been thoroughly examined for the languages Lygon [3], where resource management is translated to Boolean constraint solving, and Lolli [53, 22], which gives an algorithmic solution for a large fragment of intuitionistic linear logic. The examination of resource management in the present work derives its inspiration from the latter of the two approaches, though the situation for forward search turns out to be very different. In fact, the defining resource management problem in the backward direction turns out to be entirely absent in the forward direction. (For details, see section 3.1.)

The particular forward search strategy we use is the *inverse method* [73, 35]. The inverse method is a generalisation of resolution [102, 84, 83] that applies to a wide variety of logics, unlike resolution which only works for classical logic. Indeed, the inverse method can be seen to be logic-independent because it has very minimal requirements: a sequent calculus with the subformula property. A brief sketch of the inverse method follows. First, the given goal sequent ("query") is fixed, and initial sequents for atomic propositions that occur both as positive and negative subformulas of the goal sequent (see defn. 4.4). Next, the inference rules of the logic are specialised to the subformulas of the goal sequent such that the principal formula in all inference rules is a subformula of the goal sequent. These rules are then used to construct new sequents by matching the premisses against previously derived sequents. New sequents that are not simply instances of sequents derived earlier are themselves then used in the inference rules to derive newer sequents. Eventually, assuming the search strategy is complete, either the goal sequent is derived, or the search space is saturated and the goal sequent is found to be unprovable. The inverse method is thus a member of a general class of *saturation-based* search procedures.

The choice of the inverse method in this thesis is further motivated by the many desirable properties of forward reasoning. Prime among these properties is localized nature of forward sequents; sequents in disjoint branches of a derivation share no existential variables. In fact, the lack of multiplicative resource non-determinism in the forward direction can be seen as another aspect of locality – resource consumption is not allowed to affect disjoint branches. From an implementation standpoint, locality allows many transparent and logically motivated optimisations. For example, because search in the inverse method is free of backtracking, existential variables need not maintain local "undo" histories. A related property is that sharing of derivations is immediate in forward reasoning, and fair exploration strategies in the forward direction will generally find shorter proofs than backward reasoning. For this reason, forward reasoning often gives decision procedures for fragments of the logic that are not as easily decided in backward search. Backward search lends itself well to eager exploration strategies in the tradition of logic programming. Extending this kind of search for general theorem proving requires sophisticated loop detection and suspension algorithms. The arguments for forward reasoning are presented in more detail in chapter 4.

1.3 Contributions

A primary contribution of this thesis is to construct an inverse method theorem prover that accounts for all the connectives of first order linear logic. Although linear logic is already about nineteen years old, a satisfactory theorem prover for the full logic is lacking. There have been many attempts to create automated deduction systems for various fragments of linear and affine logic, both classical and intuitionistic, but these systems all necessitate adapting to the lack of all linear connectives. The nineteen years of experience have, therefore, seen the emergence of common idioms in encoding linear theories in fragments such as hereditary Harrop formulas that virtually all linear logic programming engines support. These idioms necessarily sacrifice the structure of proofs, for example, by selecting particular serialisations of concurrent behaviour, as is standard in continuation-passing-style and related encodings.

In this thesis, we answer the general theorem proving problem for all linear connectives, without any idiomatic commitments. Of course, any use of a general theorem prover does not preclude limiting one's use to a fragment of the logic. Indeed, it has been remarked for theories that fall in such well behaved fragments that it might be possible to improve the efficiency of the inverse method search procedure by incorporating hyper-resolution strategies; see, for example, Tammet's treatment of classical logic in the Gandalf prover [108]. Such specialised strategies have practical benefits, but they are not very satisfactory from both design and engineering. Furthermore, it is a denouncement of the versatility of the inverse method if one were simply to abandon it for a radically different procedure at times. Instead, the work in this thesis was motivated by a question posed by Pfenning at its inception: can strategies such as hyperresolution be explained *in terms of* the inverse method?

This thesis gives a substantial answer to this question. Looking at the behaviour of an inverse method prover in practice, the bottleneck always is the size of the sequent database. As the sequents in the database are considered for generation of new sequents, as the number of sequents in the database increases, it has the effect of slowing down rule applications as there are a larger number of possible premisses to check for every rule. Additionally, the dominant operation in the prover soon becomes the task of detecting when a newly constructed sequent is globally new (i.e., not subsumed by another sequent in the database). Of course, in any saturation-based approach this problem is technically unavoidable, but the impetus of design for such provers should be to reduce the size of the database.

Our answer is to combine the inverse method with the notion of focused derivations [7]. Focused derivations arose in the context of logic programming as a way of refining proof search into phases. Each phase of the search consisted either of only *asynchronous* steps where non-determinism was immaterial, or of only *synchronous* steps where key choices have to be made. Focusing was thus a way of making "big step" derivations: pairs of synchronous and asynchronous steps could be thought of as a large derived rule. In this work we make the observation that these derived inference rules constructed by focusing can also be used to do forward search in big steps. Thus, the intermediate results that are internal to the phases of a focused derivations do not have to be explicitly constructed or stored in a sequent database. This reduces the size of the sequent database, which is the main bottleneck in the inverse method. Because a focusing inverse method prover is able to make much larger inferences in much fewer steps, it is able to explore the search space much more efficiently. In our experiments, we routinely observed the focusing prover outperforming the non-focusing ("small-step") prover by several orders of magnitude.

In this thesis we reconstruct focused derivations from first principles. The resulting calculus is both simpler and more efficient than other focusing calculi that have been proposed for intuitionistic logics [57]. In particular, we highlight the important concept of focusing bias for atomic propositions and the effect the choice of bias has on the derived rules generated during focusing. In chapter 6 we show how one choice of bias gives rise to hyperresolution, a forward-chaining strategy, whereas the opposite choice gives rise to SLD-resolution, a backward-chaining strategy. Focusing bias can therefore be seen as a logical explanation for the operational notions of forward or backward chaining, and we are able to combine both operations in a seamless manner just by selecting appropriate biases for the atomic propositions. In chapter 7 we show that proper selection of focusing bias can significantly improve the performance of the prover.

The work in this thesis is supported by an implementation of a competitive theorem prover for intuitionistic first-order linear logic. Of course, the field of competition is currently sparse—no other theorem prover exists for first-order linear logic, for example—but we are able to provide evidence for the merit of the focused inverse method by internal comparisons with variant implementations. The results of these experiments have caused the author to undergo a kind of religious conversion, as he now zealously preaches that focused derivations be used as a matter of course in all future automated reasoning systems, be they backward- or forward-reasoning.

1.4 Structure of the thesis

Chapter 2 presents the background of this thesis. The backward sequent calculus for the full logic is presented in increments, including an extension with the possibility judgement of JILL [27]. The key presentational component of this chapter is the proof of cut-elimination for the full logic that is presented constructively as a computation on sequent derivations in what is now known as the "structural cut elimination" method [92]. This chapter ends with a discussion of proof-presentation as normal natural deduction proofs. Subsequent chapters build up to a forward calculi for this full logic.

Chapter 3 introduces the forward sequent calculus for the propositional fragment of the logic. Here we see the first important concept necessary for forward reasoning in linear logic– that of *weak sequents*. This forward calculus annotated with weakenable linear contexts is proven sound and complete with respect to the propositional backward sequent calculus of chapter 2. This chapter ends with a discussion of two major optimisations. The first of these is independent of the propositional nature of this logic and deals with a heuristic for handling locally affine theories. The second optimisation discusses the benefit of actively preventing redundant sequents from being constructed, but the development advanced in this chapter is only feasible in the propositional case. The propositional variant of the LI prover (i.e., LIP) uses this irredundant formulation even though the idea does not generalise to the first-order or focusing cases.

Chapter 4 presents the inverse method procedure that uses the propositional forward calculus of chapter 3. The subformula property is highlighted, and the method of specialising rules to subformulas is explained in this chapter. The primary technical contribution of this chapter has to do with the representation of sequents and the details of the lazy OTTER loop.

Chapter 5 extends the propositional calculus of earlier sections with first-order quantification. The structure of this chapter follows the "recipe" outlined in [35] by first presenting a ground version of the forward calculus, and then lifting it to a calculus with free variables and unification. Although the procedure is fairly standard, the interactions of this procedure with linear logic, particularly the additive connectives, are contributions of this work.

Chapter 6 is the key contribution of this thesis. In this chapter, a calculus of focused derivations is reconstructed from first principles. The notion of focusing bias presented in this chapter is a version of a similar observation made by Andreoli for classical linear logic [7], but was extended to intuitionistic linear logic in this work. This chapter also presents a novel proof of completeness of the focusing calculus with respect to the non-focusing calculus (of chapter 2) by means of cut-elimination. A calculus of (backward) derived inference rules is then formally extracted from this focusing calculus; subsequently the forward version of this calculus is presented and proved sound and complete with respect to the backward calculus. Finally, the details of combining derived inference rules with many premisses with the inverse method is presented. We then examine a few translations: the first of these shows that a focusing sequent calculus for natural (non-linear) logic can be translated to the linear setting while preserving the focusing structure of proofs. Finally, we explain how hyperresolution and SLD resolution arise naturally as differently biased variants of the focusing calculus.

Chapter 7 presents the implementation of the calculi of earlier chapters and the results of a number of experiments. Every variant mentioned in this thesis, including the left-biased and right-biased version of the focusing calculi, has been implemented and compared on suitable examples. As mentioned previously, the possibilities for external comparisons is limited because of the small number of provers that exist for linear logic; therefore, much of the experimental validation of the claims of this thesis comes from internal comparisons.

Chapter 8 summarises the conclusions of this thesis and briefly discusses future work.

1.5 Literature survey

1.5.1 Automated reasoning approaches based on logic programming

Resource management has a relatively long history given the age of linear and substructural logics, with the earliest identification of this issue in uniform proof search dating back to the work of Harland and Pym in 1991 [48] (see also [98]). Their approach centred on rewriting the hereditary Harrop fragment of linear logic in a clausal form suitable for resolution search. Although they stopped short of identifying the resource management problem as such, they did give an outline of a solution for multiplicative resource non-determinism, which revolved around constructing non-linear proto-proofs and a subsequent pass that reconstructs a linear proof. Central to this idea was an attempt to delay splitting the linear context as long as possible. In subsequent work they explored the implementation details of their lazy algorithm, which eventually led to an elegant formulation of uniform proof search in terms of Boolean constraints [49]. This system has since been implemented in the logic programming language Lygon [3].

Hodas and Miller took a different approach to linear logic programming [53, 56]. Instead of treating it as a restriction of general backward proof search, they set out to discover a specific solution and expose it directly in the proof theory of Lolli. In subsequent work, Cervesato *et al* [22] have extended the approach to handle resource management efficiently in the presence of the additive conjunction &. The weakening annotation described in the present work bears a strong resemblance to a similar notation in [22], although the interpretation differs considerably because of the different nature of forward search.

Linear logic programming has also been examined from the perspective of specification languages, first by Miller in the system Forum [78] and Andreoli in the system LinLog [7]. The latter work, in fact, introduced the dyadic notation for resources used extensively in the present work. More recently, Cervesato and Pfenning [24] have attempted to provide a sound type-theoretic foundation to linear logic programming in terms of extensions to the LF logical framework [51]. Watkins *et al* have further extended this line of work by giving a manifestly decidable equational theory for an extension of LF with a monad for concurrency [117, 25].

1.5.2 General theorem proving

All the systems described so far have been restrictions of classical or intuitionistic linear logic for particular domains. The general theorem-proving problem for the full classical linear logic has been investigated by Mints in the style of resolution theorem proving [83]. Mints did not identify or provide a solution to the resource management problem; indeed, his original calculus was unimplementable as a standard resolution theorem prover because it had clauses containing an undetermined collection of resources. (This problem closely resembles structural resource non-determinism identified in section 3.1.) To mitigate this problem, Mints provided a general strategy to delay uses of such clauses as late as possible, and this line was further investigated by Tammet [110], who discovered many permissible permutations of rules, making Mints's calculus implementable. However, Tammet did not present the resource management issues in isolation from particular resolution strategies.

1.5.3 Other logics for stateful systems

Approaches based on rewriting logic Rewrite systems like ELAN [2] and Maude [4] have been considered for specifications of stateful systems. In fact, Maude is powerful enough to encode LinLog [7], and has been used to give a general and logical treatment for planning domains [71]. More sophisticated approaches based on multi-set rewrite systems have been employed by Cervesato [21] to model cryptographic authentication protocols. Such systems are easily embedded in a fragment of the CLF framework [117], where the existential quantification used to model nonces in MSR is translated to a similar construct in CLF.

Multiset rewrite systems have also been studied extensively by Bozzano for his PhD thesis in the context of model checking approaches for unbounded state systems [14]. Instead of traditional approaches based on finite abstraction, using human input or more automated methods, Bozzano translates the verification problem to a the language LO [9] parametrised over constraint domains such as Herbrand universes. Bozzano extended LO with universal quantification, and gave a top down saturation-based inference mechanism that was shown to be a decision procedure for interesting domains. Bozzano and Delzanno have since explored many model checking approaches using similar methods; for a survey, see [16].

Approaches based on temporal or modal logics An increasingly popular class of system specifications has used temporal logics like TLA [76], LTL or CBL, to model the evolutionary behaviour of concurrent systems. Temporal logic allows expressive descriptions of behaviour such as "eventually always *P*" or "infinitely often *P*". Systems such as Lamport's TLA are not designed with automation as their primary aim; rather, they are intended to engage engineers in the act of writing formal specifications. Nevertheless, TLA endowed with a variant of ZF set-theory (also known as TLA+) does allow for a rudimentary kind of model checking. Insofar as specifications in TLA+ are systematically constructed in the so-called *standard form*, they resemble specifications one might as well have written in linear logic.

1.5.4 Model checking and related approaches

Model checking is the chief alternative to theorem proving. It differs from theorem proving because of a bottom-up approach of assembling efficient decision procedures for small domains, rather than performing general logical inference. Model checking is therefore inherently limited in the kinds of problems it can handle. For example, model checking is unable to handle domains with unbounded state, as is natural in such domains as communication protocols [14]. A common approach is to use a finite approximation of the problem by means of such techniques as predicate abstraction [47], which has also been used extensively for hardware model-checking [34].

Predicate abstraction forms the core of modern software model-checking systems such as SLAM [10] or BLAST [1]. The general approach taken in such systems is to abstract a system at two levels – a *specification* level and a *model* level, and the verification problem is to ensure that the model is a refinement of the specification. The art of inferring models in a tractable manner can be surprisingly subtle because of the exponential blowup in the state space in the presence of concurrency. Two methods are commonly used to combat the state explosion problem – compositional reasoning [31, 85], which attempts to break up a big problem into manageable components, and *partial order reduction* [45, 69, 90, 114]. Combinations of such methods can nowadays handle model-checking problems of the order of 10^{100} states.

Efficient model checking back-ends are, nevertheless, ultimately unsatisfactory as descriptive frameworks for concurrent systems. In the domain of concurrent processes, particularly, many abstractions have been proposed to model concurrency and communication — CSP [52, 19], CCS [81], the π-calculus [82] or Petri-nets [91]. (These abstractions all fall naturally into fragments of linear logic [25].) Combining model checking approaches with such abstractions is a non-trivial task. A promising recent approach is to capture the refinement relations, to be turned into proof-obligations for a model-checking back-end, into *behavioral types* that are exposed in the abstraction itself [63, 26]. While an interesting use of type-theory, this approach must still be regarded as a *tour de force* of existing model checking frameworks such as SLAM.

It is important to stress that theorem proving and model checking are not competitors; in fact, each has a lot to offer the other. Logical inference gives a way for model checking to exceed the the finite state limitation (as demonstrated by Bozzano [14]), and terminate exploration early for logical impossibilities. Model checking, in turn, adds a number of efficient search strategies to the arsenal of inference mechanisms in a theorem prover.

Publication list

Much of the work presented in this thesis has also appeared in the following conference publications and technical reports.

[1] Kaustuv Chaudhuri, Frank Pfenning and Greg Price. A Logical Characterization of Forward and Backward Chaining in the Inverse Method. In U. Furbach and N. Shankar, eds., *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR)*, pages 97–111, August 2006. Springer-Verlag LNAI 4130

[2] Kaustuv Chaudhuri and Frank Pfenning. Focusing the Inverse Method for Linear Logic. In L. Ong, ed., *Proceedings of Computer Science Logic*, pages 200–215, Oxford, UK, August 2005. Springer-Verlag LNCS 3634

[3] Kaustuv Chaudhuri and Frank Pfenning A Focusing Inverse Method Theorem Prover for First-Order Linear Logic. In *Proceedings of CADE-20*, pages 69–83, Tallinn, Estonia, 2005. Springer-Verlag LNAI 3632.

[4] Kaustuv Chaudhuri. An Inverse Method Theorem Prover for Linear Logic (The Propositional Fragment). Technical Report CMU-CS-03-140, Carnegie Mellon University, November 2003

Chapter 2

Intuitionistic Linear Logic

2.1 Sequent calculus

We take a foundational view of logic based on the approach laid out by Martin-Löf in his Siena lectures in 1983 [72, 94]. This view rests on a separation of the *judgements* from *propositions*. A judgement becomes *evident* when it is established by means of a *proof*. The primary judgements of natural (i.e., non-linear) logic are of the form "*A is a proposition*" and "*A is true*". To understand the meaning of a proposition, we need to understand what counts as a verification of that proposition; the inference rules characterising the truth of propositions define their meaning, as long as they satisfy certain local consistency conditions. Many aspects of this foundational reconstruction of logic are now standard features of the so-called *judgemental* philosophy of logic, usually presented in a natural deduction formalism. Instead of repeating this usual introduction, we start directly with a sequent calculus, leaving a discussion of the natural deduction formulation for purely proof-presentation purposes (sec. 2.2).

We import this judgemental view into a Gentzen-style sequent calculus for linear logic [41, 40]. In this calculus, we discard the general judgement of *truth* in favour of more basic notions of *resources* and *goals*, that is, with the judgements "*A is a resource*" and "*A is a goal*". These correspond to *hypotheses* and *conclusions*, respectively, but with the following *linearity* restriction: every resource used to construct a goal must be *consumed* exactly once. As usual, we write this as a *sequent*, with the resources listed on the left of the sequent arrow ($\Longrightarrow$), and the goal on the right.

$$u_1 : (A_1\ \mathit{res}), u_2 : (A_2\ \mathit{res}), \ldots, u_n : (A_n\ \mathit{res}) \Longrightarrow C\ \mathit{goal}$$

This sequent is read as follows: with resources A_i we can achieve goal C.

Each u_i is a label for the particular resource, and all labels are distinct. We shall write Δ for the collection of resources. Where understood, we shall elide both the label for the resource, and the judgemental label " *res*". A given sequent is taken to be evident if it has a *derivation* using the *rules of inference*. Of these rules of inference there are two kinds: *judgemental* rules (sometimes also called *structural rules*) that define the allowable structural operations on sequents, and *logical* rules that define the meaning of logical connectives.

Judgemental rules Though we have yet to specify our language of propositions, they will include a collection of *atomic* propositions. We write these propositions using lower-case letters: p, q, etc. Atomic propositions have no propositional structure, and therefore no logical rules defining their meaning. There is one judgemental principle to characterise the use of atomic propositions: an atomic resource can be consumed to obtain the same atomic goal. This we write in the form of an axiomatic *initial* rule:

$$\frac{}{p\ \mathit{res} \Longrightarrow p\ \mathit{goal}}\ \text{init.}$$

Unlike natural logics, in the purely linear logic there are no further judgemental rules regarding resources. In particular, *weakening* and *contraction* are not valid structural rules

The sequent calculus is constructed to satisfy two important principles – substitution and identity. Substitution, often known as *cut*, defines how a conclusion *A goal*, relates to uses of the the hypothesis *A res*.

$$\frac{\Delta \Longrightarrow A\ \mathit{goal} \quad \Delta', A\ \mathit{res} \Longrightarrow C\ \mathit{goal}}{\Delta, \Delta' \Longrightarrow C\ \mathit{goal}}\ \text{cut}$$

In other words, it is sound to use a goal as a resource. Dually, given any resource, the logic must be strong enough to construct a goal from it.

$$\frac{}{A\ \mathit{res} \Longrightarrow A\ \mathit{goal}}\ \text{identity}$$

These two principles can be seen as a form of global soundness and completeness of the calculus with respect to the natural deduction formulation of the logic (see sec. 2.2).

Theorem 2.1. *The "cut" and "identity" rules are admissible.*

For expository purposes, we shall present principal cases of the proof of this theorem as we present the logic. The full proof will be delayed until the formal presentation in section 2.1.5.

2.1.1 Purely linear logic

Now we turn to the linear propositions, whose meanings are given in terms of their *logical rules*. Linear propositions are built up in terms of the following grammar:

$$A, B, C, \ldots \quad ::= \quad A \otimes B \mid \mathbf{1} \mid A \oplus B \mid \mathbf{0} \mid A \multimap B \mid A \mathbin{\&} B \mid \top$$

For each connective we will have *right rules* that will define the *construction* of the proposition as goals, and *left rules* that will define the use of the proposition as resources.

Multiplicative conjunction The $\otimes$ connective (called "tensor") and its unit $\mathbf{1}$ are constructed as goals as follows:

$$\frac{\Delta \Longrightarrow A\ goal \quad \Delta' \Longrightarrow B\ goal}{\Delta, \Delta' \Longrightarrow A \otimes B\ goal}\ \otimes R \qquad \frac{}{\cdot \Longrightarrow \mathbf{1}\ goal}\ \mathbf{1}R$$

The operation Δ, Δ' denotes *multiplicative union*, that is, each of the constituents is present wholly and separately in the united context. On the left, to use $A \otimes B$ as a resource is to use both A and B as resources; to use $\mathbf{1}$ as a resource is to remove it. Thus we obtain the left rules:

$$\frac{\Delta, A\ res, B\ res \Longrightarrow C\ goal}{\Delta, A \otimes B\ res \Longrightarrow C\ goal}\ \otimes L \qquad \frac{\Delta \Longrightarrow C\ goal}{\Delta, \mathbf{1}\ res \Longrightarrow C\ goal}\ \mathbf{1}L$$

Why are these the correct rules for these connectives? To answer that, we have to look at local versions of the global soundness and completeness theorems, "cut" and "identity", respectively.

The local form of "cut" is what is called a *principal cut*. Here, we assume that (for some chosen proposition C) we have two derivations, one in which $C\ goal$ has just been constructed using a right rule, and another in which $C\ res$ has just been used with a left rule. Explicitly, for $\otimes$, we consider the derivations $\mathcal{D} :: \Delta \Longrightarrow A \otimes B\ goal$ and $\mathcal{E} :: \Delta', A\ res \Longrightarrow C\ goal$

as follows:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Delta_1 \Longrightarrow A\ goal \quad \mathcal{D}_2 :: \Delta_2 \Longrightarrow B\ goal}{\Delta_1, \Delta_2 \Longrightarrow A \otimes B\ goal}\ \otimes R$$

$$\mathcal{E} = \frac{\mathcal{E}' :: \Delta', A\ res, B\ res \Longrightarrow C\ goal}{\Delta', A \otimes B\ res \Longrightarrow C\ goal}\ \otimes L$$

A cut between these two derivations should give us a means of decomposing the cut to smaller derivations for which we assume, inductively, that cuts are admissible. For instance, we can cut (written using + indexed with the cut proposition) the derivation $\mathcal{D}_1$ and $\mathcal{E}'$ to get:

$$(\mathcal{D}_1 +_A \mathcal{E}') :: \Delta_1, \Delta', B\ res \Longrightarrow C\ goal$$

This derivation is not necessarily smaller than either $\mathcal{D}_1$ or $\mathcal{E}'$; however, we still allow a cut on this derivation with $\mathcal{D}_2$ because it has a smaller cut formula, in this case B. Thus,

$$\mathcal{D}_2 +_B (\mathcal{D}_1 +_A \mathcal{E}') :: \Delta_1, \Delta_2, \Delta' \Longrightarrow C\ goal$$

To state principal cuts even more precisely, we need a language for derivations. The right derivation $\mathcal{D}$ is constructed by multiplicatively compositing the right derivations $\mathcal{D}_1$ and $\mathcal{D}_2$; we therefore write this as

$$\otimes R(\mathcal{D}_1, \mathcal{D}_2\ ; A \otimes B)$$

Similarly, the left derivation $\mathcal{E}$ is constructed from $\mathcal{E}'$ by combining $u : (A\ res)$ and $v : (B\ res)$ to create $w : (A \otimes B\ res)$. We write this as:

$$\otimes L(u.\ v.\ \mathcal{E}'\ ; w{:}A \otimes B)$$

The labels u and v are understood as free in $\mathcal{E}'$, but bound in $u.\ v.\ \mathcal{E}'$. The label w is free in the full $\otimes L$ derivation. We use the substitution notation $[u_1/v_1, \ldots, u_n/v_n]\mathcal{D}$ to denote the simultaneous renaming of the free labels v_i in the derivation $\mathcal{D}$ with the labels u_i. Of course, for the resulting derivation to be well-formed u_i and v_j must all be distinct, and furthermore none of the u_i must already occur freely in $\mathcal{D}$. Once we have the full syntax of derivations, we shall formalise the notion of free labels (see defn. 2.16).

In each case in the syntax for derivations, we separate out the principal proposition with a semi-colon. If the principal formula is on the left, we further indicate the label of the hypothesis assigned to the principal formula; for this derivation to be well-formed, this label must not occur freely in any subderivation. The full syntax of all derivations will be presented in section 2.1.5.

The principal cut above can be written as a simple equation on derivations:

$$\otimes R(\mathcal{D}_1, \mathcal{D}_2 \,;\, A \otimes B) +_{w:A\otimes B} \otimes L(u.\, v.\, \mathcal{E} \,;\, w{:}A \otimes B) \quad = \quad \mathcal{D}_2 +_{v':B} (\mathcal{D}_1 +_{u':A} [u'/u, v'/v]\mathcal{E})$$

where u' and v' are assumed to be fresh variables, i.e.,

$$\{u', v'\} \cap \Big(\mathrm{fl}(\mathcal{D}_1) \cup \mathrm{fl}(\mathcal{D}_2) \cup \mathrm{fl}(\mathcal{E}) \cup \{u, v\}\Big) = \emptyset$$

For the unit, **1**, we obtain a very similar principal cut.

$$\mathbf{1}R +_{u:\mathbf{1}} \mathbf{1}L(\mathcal{E} \,;\, u{:}\mathbf{1}) \quad = \quad \mathcal{E}$$

Next the question of local completeness. Here we prove the identity principle inductively on the proposition, assuming it for sub-propositions.

$$\dfrac{\dfrac{\dfrac{}{A \Longrightarrow A}\,\text{i.h.} \qquad \dfrac{}{B \Longrightarrow B}\,\text{i.h.}}{A, B \Longrightarrow A \otimes B}\,\otimes R}{A \otimes B \Longrightarrow A \otimes B}\,\otimes L \qquad\qquad \dfrac{\dfrac{}{\cdot \Longrightarrow \mathbf{1}}\,\mathbf{1}R}{\mathbf{1} \Longrightarrow \mathbf{1}}\,\mathbf{1}L$$

Linear implication The linear implication $\multimap$ (sometimes called "lolli") is constructed as a goal and used as a resource in the following ways:

$$\dfrac{\Delta, A\ \mathit{res} \Longrightarrow B\ \mathit{goal}}{\Delta \Longrightarrow A \multimap B\ \mathit{goal}}\,{\multimap}R \qquad \dfrac{\Delta \Longrightarrow A\ \mathit{goal} \quad \Delta', B\ \mathit{res} \Longrightarrow C\ \mathit{goal}}{\Delta, \Delta', A \multimap B\ \mathit{res} \Longrightarrow C\ \mathit{goal}}\,{\multimap}L$$

The corresponding right and left derivations use the following syntax:

$${\multimap}R(u.\, \mathcal{D} \,;\, A \multimap B) \qquad\qquad {\multimap}L(\mathcal{E}, u.\, \mathcal{E}' \,;\, v{:}A \multimap B)$$

The principal cut is:

$${\multimap}R(u.\, \mathcal{D} \,;\, A \multimap B) +_{w:A\multimap B} {\multimap}L(\mathcal{E}, v.\, \mathcal{E}' \,;\, w{:}A \multimap B) \quad = \quad \Big(\mathcal{E} +_{u':A} [u'/u]\mathcal{D}\Big) +_{v':B} [v'/v]\mathcal{E}'$$

where, as before, the variables u' and v' are fresh, i.e., not occurring in any of the smaller derivations. The inductive case of the identity principle is:

$$\dfrac{\dfrac{\dfrac{}{A \Longrightarrow A}\,\text{i.h.} \qquad \dfrac{}{B \Longrightarrow B}\,\text{i.h.}}{A \multimap B, A \Longrightarrow B}\,{\multimap}L}{A \multimap B \Longrightarrow A \multimap B}\,{\multimap}R$$

Additive conjunction The additive conjunction & (called "with") has the following right and left rules.

$$\frac{\Delta \Longrightarrow A\ goal \quad \Delta \Longrightarrow B\ goal}{\Delta \Longrightarrow A \mathbin{\&} B\ goal}\ \&R$$

$$\frac{\Delta, A\ res \Longrightarrow C\ goal}{\Delta, A \mathbin{\&} B\ res \Longrightarrow C\ goal}\ \&L_1 \qquad \frac{\Delta, B\ res \Longrightarrow C\ goal}{\Delta, A \mathbin{\&} B\ res \Longrightarrow C\ goal}\ \&L_2$$

The corresponding syntax for right and left derivations is;

$$\&R(\mathcal{D}_1, \mathcal{D}_2\,; A \mathbin{\&} B) \qquad \&L_i(u.\ \mathcal{D}\,; w{:}A_1 \mathbin{\&} A_2) \qquad i \in \{1,2\}$$

There are two principal cuts, depending on which of the left rules was used for $A \mathbin{\&} B\ res$.

$$\begin{array}{rcl} \&R(\mathcal{D}_1, \mathcal{D}_2\,; A \mathbin{\&} B) +_{u:A\&B} \&L_1(v.\ \mathcal{E}\,; u{:}A \mathbin{\&} B) & = & \mathcal{D}_1 +_{u':A} [v'/v]\mathcal{E} \\ \&R(\mathcal{D}_1, \mathcal{D}_2\,; A \mathbin{\&} B) +_{u:A\&B} \&L_2(v.\ \mathcal{E}\,; u{:}A \mathbin{\&} B) & = & \mathcal{D}_2 +_{u':B} [v'/v]\mathcal{E} \end{array}$$

(Again, per convention, the variable v' is fresh.)

The inductive case of the identity principle is:

$$\frac{\dfrac{\dfrac{}{A \Longrightarrow A}\ \text{i.h.}}{A \mathbin{\&} B \Longrightarrow A}\ \&L_1 \qquad \dfrac{\dfrac{}{B \Longrightarrow B}\ \text{i.h.}}{A \mathbin{\&} B \Longrightarrow B}\ \&L_2}{A \mathbin{\&} B \Longrightarrow A \mathbin{\&} B}\ \&R$$

The unit of the additive conjunction is $\top$. It has a single right rule:

$$\frac{}{\Delta \Longrightarrow \top\ goal}\ \top R$$

Being a nullary case of &, it lacks a left rule. Thus the question of a principal cut doesn't arise at all for this connective. The inductive case of the identity principle is trivial:

$$\frac{}{\top\ res \Longrightarrow \top\ goal}\ \top R$$

Disjunction Disjunction $\oplus$ has the following right and left rules.

$$\frac{\Delta \Longrightarrow A\ goal}{\Delta \Longrightarrow A \oplus B\ goal}\ \oplus R_1 \qquad \frac{\Delta \Longrightarrow B\ goal}{\Delta \Longrightarrow A \oplus B\ goal}\ \oplus R_2 \qquad \frac{\Delta, A\ res \Longrightarrow C \quad \Delta, B\ res \Longrightarrow C}{\Delta, A \oplus B\ res \Longrightarrow C}\ \oplus L$$

The corresponding syntax for right and left derivations is;

$$\oplus R_i(\mathcal{D}\,; A_1 \oplus A_2) \qquad \oplus L(u.\ \mathcal{E}_1, v.\ \mathcal{E}_2\,; w{:}A \mathbin{\&} B) \qquad i \in \{1,2\}$$

Once again we obtain two principal cuts, depending on which rule was used to conclude $A \oplus B$ *goal*.

$$\oplus R_i(\mathcal{D}\,; A_1 \oplus A_2) +_{uA_1 \oplus A_2} \oplus L(v_1.\,\mathcal{E}_1, v_2.\,\mathcal{E}_2\,; u\!:\!A_1 \oplus A_2) \quad = \quad \mathcal{D} +_{v'_i A_i} [v'_i/v_i]\mathcal{E}_i \qquad i \in \{1,2\}$$

(Per convention, v'_i is fresh.) The inductive case of the identity principle is:

$$\dfrac{\dfrac{\dfrac{}{A \Longrightarrow A}\,\text{i.h.}}{A \Longrightarrow A \oplus B}\,\oplus R_1 \qquad \dfrac{\dfrac{}{B \Longrightarrow B}\,\text{i.h.}}{B \Longrightarrow A \oplus B}\,\oplus R_2}{A \oplus B \Longrightarrow A \oplus B}\,\oplus L$$

The unit of $\oplus$ is $\mathbf{0}$; it has no right rules, and a single left rule:

$$\dfrac{}{\Delta, \mathbf{0}\; res \Longrightarrow C\; goal}\,\mathbf{0}L$$

There is no principal cut, and the inductive case of the identity principle is:

$$\dfrac{}{\mathbf{0}\; res \Longrightarrow \mathbf{0}\; goal}\,\mathbf{0}L$$

2.1.2 Truth

Logical truth is recovered in this resource-aware setting in the form of a *modal categorical judgement*, that is, a sequent with no resources.

The judgement A *true* is evident if and only if $\cdot \Longrightarrow A$ *goal*.

That is, a true proposition is independent of the linear resources. Any proof of a true proposition can therefore be re-used arbitrarily often without fear of consuming any linear resources. The analogue of the truth judgement on the left of the sequent arrow, therefore, is treated as an *unrestricted resource* (written *unr*), that is, a resource that may be consumed arbitrarily often. More precisely, we define a *dyadic* sequent of the form:

$$u_1\!:\!(A_1\; unr), \ldots, u_m\!:\!(A_m\; unr) \quad ; \quad v_1\!:\!(B_1\; res), \ldots, v_n\!:\!(B_n\; res) \Longrightarrow C\; goal.$$

The ordinary sequent $A_1\; hyp, \ldots, A_n\; hyp \Longrightarrow C\; true$ of natural logic is the derived form $A_1\; unr, \ldots, A_m\; unr\,; \cdot \Longrightarrow C\; goal$; we thus obtain one ready embedding of natural logic in linear logic: ensure that no hypotheses are linear. Again, to simplify matters, we omit the hypothesis labels u, v, etc. and the judgemental labels *unr* and *res* when there is no ambiguity. We use the meta-variables Γ and Δ for the unrestricted and linear hypotheses respectively.

Judgemental rules There is a new judgemental rule to characterise the unrestricted resources: any unrestricted resource may be copied (thinking backwards) into the linear zone arbitrarily often, using the "copy" rule:

$$\frac{\Gamma, A\,;\Delta, A \Longrightarrow C}{\Gamma, A\,;\Delta \Longrightarrow C}\ \text{copy.}$$

A logician may choose to read this rule as a form of contraction: any number of copies of an unrestricted resource may be factored away.

Whenever we add a new judgement to the logic, we have to ask what it means to construct and use this new judgement. For unrestricted resources, we obtain a new case of the "cut" rule, which we write call "cut!":

$$\frac{\Gamma\,;\cdot \Longrightarrow A\ \mathit{goal} \quad \Gamma, A\ \mathit{unr}\,;\Delta \Longrightarrow C\ \mathit{goal}}{\Gamma\,;\Delta \Longrightarrow C\ \mathit{goal}}\ \text{cut!}$$

Dually, we extend the identity principle slightly by allowing any number of unrestricted hypotheses in the identity rule.

$$\frac{}{\Gamma\,;A\ \mathit{res} \Longrightarrow A\ \mathit{goal}}\ \text{identity}$$

This form of the identity principle is all we need, because $\Gamma, A\ \mathit{unr}\,;\cdot \Longrightarrow A\ \mathit{goal}$ can be derived using "copy".

$$\frac{\dfrac{}{\Gamma, A\ \mathit{unr}\,;A\ \mathit{res} \Longrightarrow A\ \mathit{goal}}\ \text{identity}}{\Gamma, A\ \mathit{unr}\,;\cdot \Longrightarrow A\ \mathit{goal}}\ \text{copy}$$

We also extend theorem 2.1 to account for the new judgement.

Theorem 2.2. *The "cut", "cut!" and "identity" rules are admissible.*

Once again, we shall present principal case of cut and a short proof of the inductive case of the identity principle as we introduce the connectives. Distinct from linear resources, the unrestricted resources may be weakened and contracted arbitrarily. In usual presentations of linear sequent calculi, these are presented as rules of inference:

$$\frac{\Gamma\,;\Delta \Longrightarrow C}{\Gamma, A\,;\Delta \Longrightarrow C}\ \text{weaken} \qquad \frac{\Gamma, A, A\,;\Delta \Longrightarrow C}{\Gamma, A\,;\Delta \Longrightarrow C}\ \text{contract}$$

However, analogously to our treatment of cut, we do not consider these rules to be judgemental, but rather engineer the logic so they are admissible.

Theorem 2.3.
The rules "weaken" and "contract" are admissible.

Truth exponential We now extend the language of propositions with a new connective, !, to internalize the *unr* and *true* judgements in terms of *res* and *goal* respectively. The right and left rules of this connective are as follows:

$$\frac{\Gamma \Longrightarrow A\ \mathit{true}}{\Gamma\,;\cdot \Longrightarrow {!A}\ \mathit{goal}}\ !R \qquad \frac{\Gamma, A\ \mathit{unr}\,;\Delta \Longrightarrow C}{\Gamma\,;\Delta, !A\ \mathit{res} \Longrightarrow C}\ !L$$

Keep in mind that the premiss of $!R$ actually is an abbreviation for $\Gamma\,;\cdot \Longrightarrow A\ \mathit{true}$. These two derivations are represented syntactically as follows.

$$!R(\mathcal{D}\,;!A) \qquad\qquad !L(u.\,\mathcal{E}\,;v\,{:}\,!A)$$

The principal cut is, then:

$$!R(\mathcal{D}\,;!A) +_{u:!A} !L(v.\,\mathcal{E}\,;u\,{:}\,!A) \quad = \quad \mathcal{D} +_{!v':A} [v'/v]\mathcal{E}$$

Per the usual convention, v' is assumed to be fresh; also $+_!$ is a representation of "cut!".

The inductive case of the identity principle is as follows:

$$\cfrac{\cfrac{\cfrac{\cfrac{}{\Gamma, A\,;A \Longrightarrow A}\ \text{i.h.}}{\Gamma, A\,;\cdot \Longrightarrow A}\ \text{copy}}{\Gamma, A\,;\cdot \Longrightarrow !A}\ !R}{\Gamma\,;!A \Longrightarrow !A}\ !L$$

Embedding intuitionistic logic To illustrate the power of the new unrestricted resource judgement, we quickly present one embedding of natural (non-linear) intuitionistic logic into linear logic. Such embeddings can actually be done in a number of ways; we will sketch Girard's original embedding [42].

Definition 2.4 (Propositional intuitionistic logic)**.** *Propositions in intuitionistic logic are either atomic, or composed out of smaller propositions using the connectives* $\wedge$, $\vee$ *and* $\supset$. *Sequents in this logic have the shape* $\Gamma \Longrightarrow_I C$ *with the following rules:*

$$\frac{}{\Gamma, p \Longrightarrow_I p}\ \text{init} \qquad \frac{\Gamma \Longrightarrow_I A \quad \Gamma \Longrightarrow_I B}{\Gamma \Longrightarrow_I A \wedge B}\ \wedge R \qquad \frac{\Gamma, A_1 \wedge A_2, A_i \Longrightarrow_I C}{\Gamma, A_1 \wedge A_2 \Longrightarrow_I C}\ \wedge L_i$$

$$\frac{\Gamma, A \Longrightarrow_I B}{\Gamma \Longrightarrow_I A \supset B} \supset R \qquad \frac{\Gamma, A \supset B \Longrightarrow_I A \quad \Gamma, A \supset B, B \Longrightarrow_I C}{\Gamma, A \supset B \Longrightarrow_I C} \supset L$$

$$\frac{\Gamma \Longrightarrow_I A_i}{\Gamma \Longrightarrow_I A_1 \vee A_2} \vee R_i \qquad \frac{\Gamma, A \vee B, A \Longrightarrow_I C \quad \Gamma, A \vee B, B \Longrightarrow_I C}{\Gamma, A \vee B \Longrightarrow_I C} \vee L$$

Definition 2.5 (Girard embedding). *The Girard embedding $(-)^o$ is defined on intuitionistic propositions hereditarily as follows:*

$$(p)^o = p \qquad (A \wedge B)^o = (A)^o \mathbin{\&} (B)^o \qquad (A \supset B)^o = !(A)^o \multimap (B)^o \qquad (A \vee B)^o = !(A)^o \oplus !(B)^o$$

Theorem 2.6 (Soundness of the Girard embedding). *If $(\Gamma)^o ; \cdot \Longrightarrow (C)^o$, then $\Gamma ; \cdot \Longrightarrow_I C$.*

Proof sketch [42]. Structural induction on the derivation of $(\Gamma)^o ; \cdot \Longrightarrow (C)^o$. □

That this embedding is complete, and furthermore, preserves the structure of proofs is extremely easy to see.

Theorem 2.7 (Completeness of the Girard embedding). *If $\Gamma \Longrightarrow_I C$, then $(\Gamma)^o ; \cdot \Longrightarrow (C)^o$.*

Proof sketch [42]. Structural induction on the derivation of $\Gamma \Longrightarrow_I C$. For every case of the final rule used in this derivation, an equivalent linear derivation can be found in the image of the embedding. The following is a characteristic example.

$$\frac{\mathcal{D}_I :: \Gamma, A \vee B, A \Longrightarrow_I C \quad \mathcal{D}'_I :: \Gamma, A \vee B, B \Longrightarrow_I C}{\Gamma, A \vee B \Longrightarrow_I C} \vee L$$

$(\Gamma)^o, (A \vee B)^o, (A)^o ; \cdot \Longrightarrow (C)^o$	i.h. on $\mathcal{D}_I$
$(\Gamma)^o, (A \vee B)^o ; !(A)^o \Longrightarrow (C)^o$	$!L$
$(\Gamma)^o, (A \vee B)^o ; !(B)^o \Longrightarrow (C)^o$	similarly for $\mathcal{D}'_I$
$(\Gamma)^o, (A \vee B)^o ; (A \vee B)^o \Longrightarrow (C)^o$	$\oplus L$
$(\Gamma)^o, (A \vee B)^o, (A \vee B)^o ; \cdot \Longrightarrow (C)^o$	copy □

There are many other possible encodings of intuitionistic logic in linear logic. A large number of them were systematically investigated by Schellinx [106]; however, his work does not cover the full spectrum of possibilities of such embeddings. For example, it is possible to make the embedding focusing-aware, as we show in section 6.4.

2.1.3 Possibility

Our conclusions thus far have been of the form *A goal*, which is not sufficient to express negation or contradiction among the hypotheses. In the usual view of negation in intuitionistic logics, contradictory hypotheses describe a condition where an actual proof of the conclusion is unnecessary. Such a view violates linearity as such, as the construction of the goal must explicitly linearly consume every resource as part of its construction. One possible approach is to define negation $\neg A$ as $A \multimap \mathbf{0}$, like in Girard's translation from intuitionistic to classical linear logic, but then we give up all pretense of linearity, because *A res* and $\neg A$ *res* can be used to construct any goal at all, destroying any linear considerations that might have been used for the rest of the hypotheses. In particular, we do not want weakening and contraction to suddenly become admissible for the linear resources if there is a contradiction.

For *linear contradiction*, therefore, we have to relax the conclusion *A goal* to allow for some additional proofs, the understanding being that a proof of *A poss* is either a proof of *A goal*, or a linear contradiction among the hypotheses. In the first case, where we actually have a proof of *A goal*, we obtain a new judgemental rule:

$$\frac{\Gamma\,;\Delta \Longrightarrow A\ \mathit{goal}}{\Gamma\,;\Delta \Longrightarrow A\ \mathit{poss}}\ \text{poss}$$

Unlike truth, this is not a definition of *A poss* because this rule is not invertible. Therefore, unlike truth, we keep *A poss* as a new judgemental form on the right hand side. We do not, however, require a matching judgemental form on the left of the sequent arrow.

The use of a possible conclusion is defined in terms of a new cut principle.

$$\frac{\Gamma\,;\Delta \Longrightarrow A\ \mathit{poss} \quad \Gamma\,;A\ \mathit{res} \Longrightarrow C\ \mathit{poss}}{\Gamma\,;\Delta \Longrightarrow C\ \mathit{poss}}\ \text{cut?}$$

The justification for this form of the cut principle is as follows. Assume *A poss*; then, there may exist a contradiction among the resources $\Gamma\,;\Delta$, in which case also *C poss*. On the other hand, we may have an actual proof of *A goal*; in this case, by the ordinary cut we get *C goal*, which also means *C poss*.

Possibility exponential The possibility judgement is internalised as the connective ? (read "why not"). Its right and left rules are as follows:

$$\frac{\Gamma\,;\Delta \Longrightarrow A\ \mathit{poss}}{\Gamma\,;\Delta \Longrightarrow {?A}\ \mathit{goal}}\ ?R \qquad \frac{\Gamma\,; A\ \mathit{res} \Longrightarrow C\ \mathit{poss}}{\Gamma\,; {?A}\ \mathit{res} \Longrightarrow C\ \mathit{poss}}\ ?L$$

The corresponding syntax for derivations is:

$$?R(\mathcal{D}\,;?A) \qquad\qquad ?L(u.\,\mathcal{E}\,;v\,;?A)$$

The principal cut for this connective is:

$$?R(\mathcal{D}\,;?A) +_{u:?A} ?L(v.\,\mathcal{E}\,;u{:}?A) \quad = \quad \mathcal{D} +_{?v':A} [v'/v]\mathcal{E}$$

Here, we use $+_?$ to stand for uses of "cut?". The label v' is fresh, per the usual convention.

Finally, the inductive case of the identity principle is:

$$\dfrac{\dfrac{\dfrac{\overline{\Gamma\,;A\ \mathit{res} \Longrightarrow A\ \mathit{goal}}\ \text{i.h.}}{\Gamma\,;A\ \mathit{res} \Longrightarrow A\ \mathit{poss}}\ \text{poss}}{\Gamma\,;{?A}\ \mathit{res} \Longrightarrow A\ \mathit{poss}}\ ?L}{\Gamma\,;{?A}\ \mathit{res} \Longrightarrow {?A}\ \mathit{goal}}\ ?R$$

Theorem 2.8. *The "cut", "cut!", "cut?" and "identity" rules are admissible.*

Linear contradiction What remains is to define linear contradiction in terms of this new possibility judgement. Recall that there is no right rule for $\mathbf{0}$ *goal*, so the only way to prove it is if the hypotheses contain a contradiction; however, the problem with $\mathbf{0}$ *res* is that it destroys linearity of the resources. Using the "poss" and $?R$ rules, we can easily get from $\mathbf{0}$ *goal* to $?\,\mathbf{0}$ *goal*. Consider $?\,\mathbf{0}$ *res*. The only way to use this resource with $?L$ is to force the linear zone to be empty except for the singleton resource $?\,\mathbf{0}$ *res*. Thus, even though $\mathbf{0}$ *res* can break linearity, this property does not transfer over to $?\,\mathbf{0}$ *res*, thus giving us a means of restricting the "reach" of the contradiction. We thus obtain a way to define a linear negation: $\neg A$ defined as $A \multimap ?\,\mathbf{0}$.

The important fact to note about this linear negation is that from A *res* and $\neg A$ *res* we can conclude any C *poss*.

$$\dfrac{\overline{\Gamma\,;A\ \mathit{res} \Longrightarrow A\ \mathit{goal}}\ \text{identity} \qquad \dfrac{\overline{\Gamma\,;\mathbf{0}\ \mathit{res} \Longrightarrow C\ \mathit{poss}}\ \mathbf{0}L}{\Gamma\,;?\,\mathbf{0}\ \mathit{res} \Longrightarrow C\ \mathit{poss}}\ ?L}{\Gamma\,;A\ \mathit{res}, \neg A\ \mathit{res} \Longrightarrow C\ \mathit{poss}}\ {\multimap}L$$

It would not be possible to conclude $\Gamma\,;\Delta, A\ \mathit{res}, \neg A\ \mathit{res} \Longrightarrow C\ \mathit{poss}$ for a non-empty Δ, however. We thus obtain our required notion of contradiction that preserves linearity.

Laxity There is a closely related judgement of laxity that has arisen in the Concurrent Logical Framework (CLF) [117]. This judgement, $A\ \mathit{lax}$ is constructed out of *monadic* rather than *modal* considerations, and can be seen as a slightly more permissive form of possibility. Like possibility, this judgement is also defined in terms of its cut principle:

$$\frac{\Gamma\,;\Delta \Longrightarrow A\ \mathit{lax} \quad \Gamma\,;\Delta', A\ \mathit{res} \Longrightarrow C\ \mathit{lax}}{\Gamma\,;\Delta, \Delta' \Longrightarrow C\ \mathit{lax}}\ \{\mathrm{cut}\}$$

This judgement thus lacks sufficient strength to define linear contradiction. However, it has a practical application as a means of "staging" the computations in a monadic style. The laxity judgement is internalized as a connective, $\{A\}$, with the following judgemental, left and right rules.

$$\frac{\Gamma\,;\Delta \Longrightarrow A\ \mathit{goal}}{\Gamma\,;\Delta \Longrightarrow A\ \mathit{lax}}\ \mathrm{lax} \qquad \frac{\Gamma\,;\Delta \Longrightarrow A\ \mathit{lax}}{\Gamma\,;\Delta \Longrightarrow \{A\}\ \mathit{goal}}\ \{R\} \qquad \frac{\Gamma\,;\Delta \Longrightarrow \{A\}\ \mathit{goal} \quad \Gamma\,;\Delta', A\ \mathit{res} \Longrightarrow C\ \mathit{lax}}{\Gamma\,;\Delta, \Delta' \Longrightarrow C\ \mathit{lax}}\ \{L\}$$

In this thesis, we shall not pursue the development of the laxity judgement in any more detail, favoring the possibility judgement instead. All statements made about the possibility judgement can be readily adapted for the laxity judgement.

Embedding classical linear logic Girard has shown that classical linear logic is more expressive than intuitionistic (non-linear) logic by giving a means of embedding the latter into the former [42]. In this section we briefly sketch an embedding of classical linear logic into intuitionistic linear logic, demonstrating the expressiveness of the possibility judgement. There are several well known translations from classical to intuitionistic logics, the most well known of them perhaps being the double-negation translation [38]. We use this idea in our case, using the definition of linear falsehood, $?\,\mathbf{0}$, from the previous section.

First, we briefly sketch the classical sequent calculus, which we present in a two-sided dyadic fashion similar to the Girard's logic of unity (LU) [43]. The propositional connectives include all the intuitionistic linear connectives, and in addition the multiplicative disjunction $\parr$, its unit $\bot$, and the "why not" exponential $?$. Classical sequents are written as $\Gamma\,;\Delta \Longrightarrow_C \Omega\,;\Psi$ where Δ and Ω are linear and Γ and Ψ are unrestricted. The rules are shown below.

Judgemental

$$\frac{}{\Gamma\,;p \Longrightarrow_C p\,;\Psi}\ \text{init}$$

Multiplicative

$$\frac{\Gamma\,;\Delta \Longrightarrow_C \Omega, A\,;\Psi \quad \Gamma\,;\Delta' \Longrightarrow_C \Omega', B\,;\Psi}{\Gamma\,;\Delta,\Delta' \Longrightarrow_C \Omega,\Omega', A\otimes B\,;\Psi}\ \otimes R \qquad \frac{\Gamma\,;\Delta, A, B \Longrightarrow_C \Omega\,;\Psi}{\Gamma\,;\Delta, A\otimes B \Longrightarrow_C \Omega\,;\Psi}\ \otimes L$$

$$\frac{}{\Gamma\,;\cdot \Longrightarrow_C \mathbf{1}\,;\Psi}\ \mathbf{1}R \qquad \frac{\Gamma\,;\Delta \Longrightarrow_C \Omega\,;\Psi}{\Gamma\,;\Delta,\mathbf{1} \Longrightarrow_C \Omega\,;\Psi}\ \mathbf{1}L$$

$$\frac{\Gamma\,;\Delta \Longrightarrow_C \Omega, A, B\,;\Psi}{\Gamma\,;\Delta \Longrightarrow_C \Omega, A \parr B\,;\Psi}\ \parr R \qquad \frac{\Gamma\,;\Delta, A \Longrightarrow_C \Omega\,;\Psi \quad \Gamma\,;\Delta', B \Longrightarrow_C \Omega'\,;\Psi}{\Gamma\,;\Delta,\Delta', A \parr B \Longrightarrow_C \Omega,\Omega'\,;\Psi}\ \parr L$$

$$\frac{\Gamma\,;\Delta \Longrightarrow_C \Omega\,;\Psi}{\Gamma\,;\Delta \Longrightarrow_C \Omega,\bot\,;\Psi}\ \bot R \qquad \frac{}{\Gamma;\bot \Longrightarrow_C \cdot\,;\Psi}\ \bot L$$

$$\frac{\Gamma\,;\Delta, A \Longrightarrow_C \Omega, B\,;\Psi}{\Gamma\,;\Delta \Longrightarrow_C \Omega, A \multimap B\,;\Psi}\ \multimap L \qquad \frac{\Gamma\,;\Delta \Longrightarrow \Omega, A\,;\Psi \quad \Gamma\,;\Delta', B \Longrightarrow \Omega'\,;\Psi}{\Gamma\,;\Delta,\Delta', A \multimap B \Longrightarrow_C \Omega,\Omega'\,;\Psi}\ \multimap R$$

$$\frac{\Gamma\,;\Delta, A \Longrightarrow_C \Omega\,;\Psi}{\Gamma\,;\Delta \Longrightarrow_C \Omega,\neg A\,;\Psi}\ \neg R \qquad \frac{\Gamma\,;\Delta \Longrightarrow_C \Omega, A\,;\Psi}{\Gamma\,;\Delta,\neg A \Longrightarrow_C \Omega\,;\Psi}\ \neg L$$

Additive

$$\frac{\Gamma\,;\Delta \Longrightarrow_C \Omega, A\,;\Psi \quad \Gamma\,;\Delta \Longrightarrow_C \Omega, B\,;\Psi}{\Gamma\,;\Delta \Longrightarrow_C \Omega, A \,\&\, B\,;\Psi}\ \&R \qquad \frac{\Gamma\,;\Delta, A_i \Longrightarrow_C \Omega\,;\Psi}{\Gamma\,;\Delta, A_1 \,\&\, A_2 \Longrightarrow_C \Omega\,;\Psi}\ \&L_i$$

$$\frac{\Gamma\,;\Delta \Longrightarrow_C \Omega, A_i\,;\Psi}{\Gamma\,;\Delta \Longrightarrow_C \Omega, A_1 \oplus A_2\,;\Psi}\ \oplus R_i \qquad \frac{\Gamma\,;\Delta, A \Longrightarrow_C \Omega\,;\Psi \quad \Gamma\,;\Delta, B \Longrightarrow_C \Omega\,;\Psi}{\Gamma\,;\Delta, A \,\&\, B \Longrightarrow_C \Omega\,;\Psi}\ \oplus L$$

$$\frac{}{\Gamma\,;\Delta \Longrightarrow_C \Omega,\top\,;\Psi}\ \top R \qquad \frac{}{\Gamma\,;\Delta,\mathbf{0} \Longrightarrow_C \Omega\,;\Psi}\ \mathbf{0}L$$

Exponential

$$\frac{\Gamma\,;\cdot \Longrightarrow_C A\,;\Psi}{\Gamma\,;\cdot \Longrightarrow_C\, !A\,;\Psi}\ !R \qquad \frac{\Gamma, A\,;\Delta \Longrightarrow_C \Omega\,;\Psi}{\Gamma\,;\Delta, !A \Longrightarrow_c \Omega\,;\Psi}\ !L$$

$$\frac{\Gamma\,;\Omega \Longrightarrow_C \Omega\,;\Psi, A}{\Gamma\,;\Delta \Longrightarrow_C \Omega, ?A\,;\Psi}\ ?R \qquad \frac{\Gamma\,;A \Longrightarrow_C \cdot\,;\Psi}{\Gamma\,;?A \Longrightarrow_c \cdot\,;\Psi}\ ?L$$

This presentation of the calculus has the following nice property: weakening and contraction of the unrestricted resources are structural theorems, and cut is admissible.

Theorem 2.9 (Structural theorems)**.**

1. *If* $\Gamma\,;\Delta \Longrightarrow_C \Omega\,;\Psi$*, then* $\Gamma,\Gamma'\,;\Delta \Longrightarrow_C \Omega\,;\Psi,\Psi'$*. (weakening)*
2. *If* $\Gamma, A, A\,;\Delta \Longrightarrow_C \Omega\,;\Psi$*, then* $\Gamma, A\,;\Delta \Longrightarrow_C \Omega\,;\Psi$*. Similarly, if* $\Gamma\,;\Delta \Longrightarrow_C \Omega\,;\Psi, A, A$*, then* $\Gamma\,;\Delta \Longrightarrow_C \Omega\,;\Psi, A$*. (contraction)*
3. *The following are cases of cut.*
   (a) *If* $\Gamma\,;\Delta \Longrightarrow_C \Omega, A\,;\Psi$ *and* $\Gamma\,;\Delta', A \Longrightarrow_C \Omega'\,;\Psi$*, then* $\Gamma\,;\Delta,\Delta' \Longrightarrow_C \Omega,\Omega'\,;\Psi$*.*
   (b) *If* $\Gamma\,;\cdot \Longrightarrow_C A\,;\Omega$ *and* $\Gamma, A\,;\Delta \Longrightarrow \Omega\,;\Psi$*, then* $\Gamma\,;\Delta \Longrightarrow_C \Omega\,;\Psi$*.*
   (c) *If* $\Gamma\,;\Delta \Longrightarrow_C \Omega\,;\Psi, A$ *and* $\Gamma\,;A \Longrightarrow \cdot\,;\Psi$*, then* $\Gamma\,;\Delta \Longrightarrow_C \Omega\,;\Psi$*.*

Proof. Weakening and contraction are shown by induction on the given derivations. Cut is shown by lexicographic induction on the three forms of cut. The details of these proofs can be found in [93]. □

Being a classical logic, this sequent calculus admits more proofs than the corresponding intuitionistic sequent calculus on the intuitionistic subset.

Definition 2.10 (Intuitionistic restriction)**.** *The* intuitionistic fragment *of the classical sequent calculus, written using the sequent arrow* $\Longrightarrow_{CI}$*, is that fragment of the classical sequent calculus that uses only the connectives* $\{\otimes, \mathbf{1}, \&, \top, \oplus, \mathbf{0}, !\}$ *and has a singleton right-hand side, i.e., of the form* $A\,;\cdot$ *or* $\cdot\,;A$*.*

Theorem 2.11 (Intuitionistic linear logic is a fragment of classical linear logic)**.**

1. $\Gamma\,;\Delta \Longrightarrow_{CI} A\,;\cdot$ *if and only if* $\Gamma\,;\Delta \Longrightarrow A$ *goal.*
2. *If* $\Gamma\,;\Delta \Longrightarrow_{CI} \cdot\,;A$ *if and only if* $\Gamma\,;\Delta \Longrightarrow A$ *poss.*

Proof. We use structural induction on the given derivation in either case. □

For the double negation translation we make a syntactic definition of negation, and then translate classical propositions into intuitionistic propositions, using the defined negation to handle the classical connectives that are not present in the image.

Definition 2.12 (Double negation translation)**.** *Write $\sim A$ for $A \multimap ?\mathbf{0}$. The translation of classical to intuitionistic linear propositions, written $[\![-]\!]$, is as follows:*

$$
\begin{array}{cc}
\multicolumn{2}{c}{[\![p]\!] = p} \\
[\![A \otimes B]\!] = \sim\sim [\![A]\!] \otimes \sim\sim [\![B]\!] & [\![\mathbf{1}]\!] = \mathbf{1} \\
[\![A \parr B]\!] = \sim (\sim [\![A]\!] \otimes \sim [\![B]\!]) & [\![\bot]\!] = \sim \mathbf{1} \\
[\![A \multimap B]\!] = \sim\sim [\![A]\!] \multimap \sim\sim [\![B]\!] & [\![\neg A]\!] = \sim\sim [\![A]\!] \multimap \sim \mathbf{1} \\
[\![A \,\&\, B]\!] = \sim\sim [\![A]\!] \,\&\, \sim\sim [\![B]\!] & [\![\top]\!] = \top \\
[\![A \oplus B]\!] = \sim\sim [\![A]\!] \oplus \sim\sim [\![B]\!] & [\![\mathbf{0}]\!] = \mathbf{0} \\
[\![!\,A]\!] = !\sim\sim [\![A]\!] & [\![?\,A]\!] = \sim !\sim [\![A]\!]
\end{array}
$$

This definition is generalised to collections of propositions, contexts, and sequents in the usual point-wise manner.

The only novelty in this definition is the use of ! in the translation of $?A$ rather than the intuitionistic ? modal operator. The reason for this choice is that the semantics of the classical ? do not match up precisely with that of the intuitionistic case because of the additional modal nature in the latter.

The following is the key lemma necessary for proving the embedding theorem.

Lemma 2.13.

1. *If $\Gamma ; \Delta \Longrightarrow A$ if and only if $\Gamma ; \Delta, \sim A \Longrightarrow ?\mathbf{0}$.*
2. *$\Gamma ; \Delta, A \Longrightarrow ?\mathbf{0}$ if and only if $\Gamma ; \Delta, \sim\sim A \Longrightarrow ?\mathbf{0}$.*

Proof. For (1) use $\multimap L$ with the given sequent and the sequent $\Gamma ; ?\mathbf{0} \Longrightarrow ?\mathbf{0}$ (using the identity principle) as premisses. For (2), in the forward direction use $\multimap R$ and part (1); in the reverse direction note that $\Gamma ; A \Longrightarrow \sim\sim A$ is derivable:

$$
\cfrac{\cfrac{\cfrac{}{\Gamma ; A \Longrightarrow A}\ \text{identity}}{\Gamma ; A, \sim A \Longrightarrow ?\mathbf{0}}\ \text{part (1)}}{\Gamma ; A \Longrightarrow \sim\sim A}\ \multimap R
$$

Then apply cut (theorems 2.8 and 2.21). □

Given a context Δ, we represent by $\sim \Delta$ the context where every proposition in Δ is affixed with $\sim$.

Theorem 2.14 (Preservation). *If* $\Gamma\,;\Delta \Longrightarrow_C \Omega\,;\Psi$, *then* $[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;[\![\Delta]\!]\,,\sim[\![\Omega]\!] \Longrightarrow\, ?\,\mathbf{0}$.

Proof. By structural induction on the derivation $C :: \Gamma\,;\Delta \Longrightarrow_C \Omega\,;\Psi$. The following are a few representative cases.

Case $C = \dfrac{}{\Gamma\,;p \Longrightarrow_C p\,;\Psi}$ init.

$$
\begin{array}{ll}
[\![\Gamma]\!]\,,\sim[\![P]\!]\,;p \Longrightarrow p & \text{“init”}\\
[\![\Gamma]\!]\,,\sim[\![P]\!]\,;p,\sim p \Longrightarrow\, ?\,\mathbf{0} & \text{lem. 2.13}
\end{array}
$$

Case The last rule in C is a multiplicative rule, say:

$$C = \frac{C_1 :: \Gamma\,;\Delta \Longrightarrow_C \Omega, A\,;\Psi \quad C_2 :: \Gamma\,;\Delta' \Longrightarrow_C \Omega', B\,;\Psi}{\Gamma\,;\Delta,\Delta' \Longrightarrow_C \Omega,\Omega', A\otimes B\,;\Psi}\otimes R$$

$$
\begin{array}{ll}
[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;[\![\Delta]\!]\,,\sim[\![\Omega]\!]\,,\sim[\![A]\!] \Longrightarrow\, ?\,\mathbf{0} & \text{i.h. on } C_1\\
[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;[\![\Delta]\!]\,,\sim[\![\Omega]\!] \Longrightarrow\sim\sim[\![A]\!] & \multimap R\\
[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;[\![\Delta']\!]\,,\sim[\![\Omega']\!] \Longrightarrow\sim\sim[\![B]\!] & \text{similarly for } C_2\\
[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;[\![\Delta,\Delta']\!]\,,\sim[\![\Omega,\Omega']\!] \Longrightarrow [\![A\otimes B]\!] & \otimes R\\
[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;[\![\Delta,\Delta']\!]\,,\sim[\![\Omega,\Omega']\!]\,,\sim[\![A\otimes B]\!] \Longrightarrow\, ?\,\mathbf{0} & \text{lem. 2.13}
\end{array}
$$

Case The principal formula in the last rule in C uses a connective specific to classical linear logic. For example, consider $\perp R$:

$$C = \frac{C' :: \Gamma\,;\Delta \Longrightarrow_C \Omega\,;\Psi}{\Gamma\,;\Delta \Longrightarrow_C \Omega,\perp\,;\Psi}\perp R.$$

$$
\begin{array}{ll}
[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;[\![\Delta]\!]\,,\sim[\![\Omega]\!] \Longrightarrow\, ?\,\mathbf{0} & \text{i.h. on } C'\\
[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;[\![\Delta]\!]\,,\sim[\![\Omega]\!]\,,\mathbf{1} \Longrightarrow\, ?\,\mathbf{0} & \text{i.h. on } \mathbf{1}L\\
[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;[\![\Delta]\!]\,,\sim[\![\Omega]\!] \Longrightarrow [\![\perp]\!] & \multimap R\\
[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;[\![\Delta]\!]\,,\sim[\![\Omega]\!]\,,\sim[\![\perp]\!] \Longrightarrow\, ?\,\mathbf{0} & \text{lem. 2.13}
\end{array}
$$

Case The principal formula in the last rule of C uses the classical ?. For example:

$$C = \frac{C' :: \Gamma\,;A \Longrightarrow \cdot\,;\Psi}{\Gamma\,;?\,A \Longrightarrow \cdot\,;\Psi}\,?\,L$$

$$
\begin{array}{ll}
[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;[\![A]\!] \Longrightarrow\, ?\,\mathbf{0} & \text{i.h. on } C'\\
[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;\cdot \Longrightarrow\sim[\![A]\!] & \multimap R\\
[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;\cdot \Longrightarrow\,!\sim[\![A]\!] & !\,R\\
[\![\Gamma]\!]\,,\sim[\![\Psi]\!]\,;[\![?\,A]\!] \Longrightarrow\, ?\,\mathbf{0} & \text{lem. 2.13}
\end{array}
$$

The case for $?\,L$ is similar. □

Thus, every classical proof has an interpretation as an intuitionistic proof. To complete the embedding, we need to show that it is meaning-preserving, i.e., that every intuitionistically valid sequent in the image of the translation is also classically valid.

Theorem 2.15 (Soundness). *If* $[\![\Gamma]\!] \,, \sim [\![\Psi]\!] \,;\, [\![\Delta]\!] \,, \sim [\![\Omega]\!] \Longrightarrow \,?\mathbf{0}$ *then* $\Gamma \,;\, \Delta \Longrightarrow_C \Omega \,;\, \Psi$.

Proof sketch. Since every intuitionistic proof is also a classical proof, by lem. 2.13 we conclude that $[\![\Gamma]\!] \,, \sim [\![\Psi]\!] \,;\, [\![\Delta]\!] \,, \sim [\![\Omega]\!] \Longrightarrow_C \,?\mathbf{0} \,;\, \cdot$. We then use the fact that classically $?0 \equiv \bot$, i.e., $\sim A \equiv \neg A$ to conclude that $[\![\Gamma]\!] \,;\, [\![\Delta]\!] \Longrightarrow_C [\![\Omega]\!] \,;\, [\![\Psi]\!]$. Finally, we note that the $[\![-]\!]$ translation is an equivalence for classical linear logic (i.e., both $\cdot \,;\, [\![A]\!] \Longrightarrow_C A \,;\, \cdot$ and $\cdot \,;\, A \Longrightarrow_C [\![A]\!] \,;\, \cdot$ may be shown). □

One direct consequence of this embedding is that we can reason about classical linear theories in translation into our intuitionistic setting. The proofs that we obtain in this fashion are not only intuitionistic proof but also manifestly classically valid also because the translation is an equivalence in classical logic. However, intuitionistic proof search in the image of the translation may not be as efficient as a direct classical proof search because the space of possible proofs is much larger in the latter case. In principle, however, we lose no expressivity in restricting ourselves to intuitionistic linear logic extended with the possibility judgement.

The translation outlined in this section is a specific instance of the parametric double-negation translation due to Friedman [38], extended to the linear setting, where the parameter is instantiated to be $?\mathbf{0}$. In a recent work [27] we have extended Friedman's idea to linear logic and shown that with choices of the parameter other than $?\mathbf{0}$ yield surprising results. For example, the linear MIX rules due to Girard [42] can be modeled in the translation by selecting $\mathbf{1}$ for the parameter, giving a judgemental explanation for a calculus that supports such rules.

2.1.4 First-order quantification

The language of propositions thus far has been propositional. In this section we shall extend them with first-order quantification ($\forall$ and $\exists$). The quantifiers range over the following simple language of untyped terms.

$$\text{(terms)} \qquad s, t, \ldots \quad ::= \quad x \quad | \quad f(t_1, t_2, \ldots, t_n)$$

where x ranges over a countably infinite set of *variables*, and f over a collection of *function symbols*. As usual, constants are treated as nullary functions. Atomic predicates are extended to be predicates over these terms; that is, they are of the form $p(t_1, t_2, \ldots, t_n)$. A list of terms will be written using a vector notation, $\vec{t}$.

Initial sequents are of the form $\Gamma\,;\Delta\,; p(\vec{t})\ \mathit{res} \Longrightarrow p(\vec{t})\ \mathit{goal}$. The right and left rules for the quantifiers are as follows:

$$\dfrac{\mathcal{D} :: \Gamma\,;\Delta \Longrightarrow [a/x]A\ \mathit{goal}}{\forall R(a.\ \mathcal{D}\,; \forall x.A) :: \Gamma\,;\Delta \Longrightarrow \forall x.A\ \mathit{goal}}\ \forall R^a \qquad \dfrac{\mathcal{D} :: \Gamma\,;\Delta, u\!:\![t/x]A\ \mathit{res} \Longrightarrow J}{\forall L(u.\ \mathcal{D}, t\,; v\!:\!\forall x.A) :: \Gamma\,;\Delta, v\!:\!\forall x.A\ \mathit{res} \Longrightarrow J}\ \forall L$$

$$\dfrac{\mathcal{D} :: \Gamma\,;\Delta \Longrightarrow [t/x]A\ \mathit{goal}}{\exists R(\mathcal{D}, t\,; \exists x.A) :: \Gamma\,;\Delta \Longrightarrow \exists x.A\ \mathit{goal}}\ \exists R \qquad \dfrac{\mathcal{D} :: \Gamma\,;\Delta, u\!:\![a/x]A\ \mathit{res} \Longrightarrow J}{\exists L(a.\ u.\ \mathcal{D}\,; v\!:\!\exists x.A) :: \Gamma\,;\Delta, v\!:\!\exists x.A\ \mathit{res} \Longrightarrow J}\ \exists L^a$$

For $\forall R$ and $\exists L$, the term a represents a *parameter*, i.e., a variable that appears nowhere in the conclusion of the rule.

The principal cuts are immediate for the quantifiers.

$$\begin{aligned}
\forall R(a.\ \mathcal{D}\,; \forall x.A) +_{u:\forall x.A} \forall L(v.\ \mathcal{E}, t\,; u\!:\!\forall x.A) \quad &= \quad [t/a]\mathcal{D} +_{v':[t/x]A} [v'/v]\mathcal{E} \\
\exists R(\mathcal{D}, t\,; \exists x.A) +_{u:\exists x.A} \exists L(a.\ v.\ \mathcal{E}\,; u\!:\!\exists x.A) \quad &= \quad \mathcal{D} +_{v':[t/x]A} [v'/v][t/a]\mathcal{E}
\end{aligned}$$

Per usual convention, the label v' is taken to be fresh. The inductive cases of the identity principle are straightforward.

$$\dfrac{\dfrac{\overline{\Gamma\,; [a/x]A \Longrightarrow [a/x]A}\ \text{i.h.}}{\Gamma\,; \forall x.A \Longrightarrow [a/x]A}\ \forall L}{\Gamma\,; \forall x.A \Longrightarrow \forall x.A}\ \forall R^a \qquad \dfrac{\dfrac{\overline{\Gamma\,; [a/x]A \Longrightarrow [a/x]A}\ \text{i.h.}}{\Gamma\,; [a/x]A \Longrightarrow \exists x.A}\ \exists R}{\Gamma\,; \exists x.A \Longrightarrow \exists x.A}\ \exists L^a$$

2.1.5 Summary of the formal system

Propositions are formed out of the following grammar.

$$\begin{array}{llll}
\text{(terms)} & s, t, \ldots & ::= x & \text{(variables)} \\
 & & \ \ \mid\ f(t_1, t_2, \ldots, t_n) & \text{(functions)} \\
\text{(propositions)} & A, B, \ldots & ::= p(t_1, t_2, \ldots, t_n) & \text{(atomic)} \\
 & & \ \ \mid\ A \otimes B \mid \mathbf{1} & \text{(multiplicative conjunction and unit)}
\end{array}$$

$$
\begin{array}{lll}
| & A \multimap B & \text{(linear implication)} \\
| & A \mathbin{\&} B \mid \top & \text{(additive conjunction and unit)} \\
| & A \oplus B \mid \mathbf{0} & \text{(additive disjunction and unit)} \\
| & !A \mid ?A & \text{(exponential)} \\
| & \forall x.A \mid \exists x.A & \text{(ordinary quantifiers)}
\end{array}
$$

Sequents with their derivations have the following form:

$$\mathcal{D} :: \underbrace{u_1 : (A_1\ \mathit{unr}), \ldots, u_m : (A_m\ \mathit{unr})}_{\Gamma} ; \underbrace{v_1 : (B_1\ \mathit{res}), \ldots, v_n : (B_n\ \mathit{res})}_{\Delta} \Longrightarrow \underbrace{\begin{cases} C\ \mathit{goal}. \\ C\ \mathit{poss}. \end{cases}}_{J}$$

The judgemental labels *unr*, *res* and *goal* are suppressed except when relevant. $\mathcal{D}$ is the derivation, whose grammar will be presented together with the rules of the calculus.

Judgemental rules

$$\frac{}{\mathrm{init}(u : p(\vec{t})) :: \Gamma ; u : p(\vec{t}) \Longrightarrow p(\vec{t})}\ \mathrm{init}$$

$$\frac{\mathcal{D} :: \Gamma, u : A ; \Delta, v : A \Longrightarrow J}{\mathrm{copy}(u.\ \mathcal{D} ; v{:}A) :: \Gamma, u : A ; \Delta \Longrightarrow J}\ \mathrm{copy} \qquad \frac{\mathcal{D} :: \Gamma ; \Delta \Longrightarrow A\ \mathit{goal}}{\mathrm{poss}(\mathcal{D}) :: \Gamma ; \Delta \Longrightarrow A\ \mathit{poss}}\ \mathrm{poss}$$

Multiplicative rules

$$\frac{\mathcal{D} :: \Gamma ; \Delta \Longrightarrow A \quad \mathcal{D}' :: \Gamma ; \Delta' \Longrightarrow B}{\otimes R(\mathcal{D}, \mathcal{D}' ; A \otimes B) :: \Gamma ; \Delta, \Delta' \Longrightarrow A \otimes B}\ \otimes R$$

$$\frac{\mathcal{D} :: \Delta, u : A, v : B \Longrightarrow J}{\otimes L(u.\ v.\ \mathcal{D} ; w : A \otimes B) :: \Delta, w : A \otimes B \Longrightarrow C}\ \otimes L$$

$$\frac{}{\mathbf{1}R :: \Gamma ; \cdot \Longrightarrow \mathbf{1}}\ \mathbf{1}R \qquad \frac{\mathcal{D} :: \Gamma ; \Delta \Longrightarrow J}{\mathbf{1}L(\mathcal{D} ; u : \mathbf{1}) :: \Gamma ; \Delta, u{:}\mathbf{1} \Longrightarrow J}\ \mathbf{1}L$$

$$\frac{\mathcal{D} :: \Gamma ; \Delta, u : A \Longrightarrow B}{\multimap R(u.\ \mathcal{D} ; A \multimap B) :: \Gamma ; \Delta \Longrightarrow A \multimap B}\ \multimap R$$

$$\frac{\mathcal{D} :: \Gamma ; \Delta \Longrightarrow A \quad \mathcal{D}' :: \Gamma ; \Delta', u : B \Longrightarrow J}{\multimap L(\mathcal{D}, u.\ \mathcal{D}' ; v{:}A \multimap B) :: \Gamma ; \Delta, \Delta', v : A \multimap B \Longrightarrow J}\ \multimap L$$

Additive rules

$$\frac{\mathcal{D} :: \Gamma ; \Delta \Longrightarrow A \quad \mathcal{D}' :: \Gamma ; \Delta \Longrightarrow B}{\&R(\mathcal{D}, \mathcal{D}' ; A \,\&\, B) :: \Gamma ; \Delta \Longrightarrow A \,\&\, B}\ \&R$$

$$\frac{\mathcal{D} :: \Gamma ; \Delta, u : A_i \Longrightarrow J}{\&L_i(u.\ \mathcal{D} ; v : A_1 \,\&\, A_2) :: \Gamma ; \Delta, u : A_1 \,\&\, A_2 \Longrightarrow C}\ \&L_i \qquad i \in \{1,2\}$$

$$\frac{\mathcal{D} :: \Gamma ; \Delta \Longrightarrow A}{\oplus R_i(\mathcal{D} ; A \oplus B) :: \Gamma ; \Delta \Longrightarrow A \oplus B}\ \oplus R_1 \qquad i \in \{1,2\}$$

$$\frac{\mathcal{D} :: \Gamma ; \Delta, u : A \Longrightarrow C \quad \mathcal{D}' :: \Gamma ; \Delta, v : B \Longrightarrow J}{\oplus L(u.\ \mathcal{D}, v.\ \mathcal{D}' ; w{:}A \oplus B) :: \Gamma ; \Delta, w : A \oplus B \Longrightarrow J}\ \oplus L$$

$$\frac{}{\top R :: \Gamma ; \Delta \Longrightarrow \top}\ \top R \qquad \frac{}{\mathbf{0}L(u{:}\mathbf{0}) :: \Gamma ; \Delta, u : \mathbf{0} \Longrightarrow J}\ \mathbf{0}L$$

Exponentials

$$\frac{\mathcal{D} :: \Gamma \Longrightarrow A}{!R(\mathcal{D} ; !A) :: \Gamma ; \cdot \Longrightarrow !A}\ !R \qquad \frac{\mathcal{D} :: \Gamma, u : A ; \Delta \Longrightarrow J}{!L(u.\ \mathcal{D} ; v{:}!A) :: \Gamma ; \Delta, v : !A \Longrightarrow J}\ !L$$

$$\frac{\mathcal{D} :: \Gamma ; \Delta \Longrightarrow A\ \mathit{poss}}{?R(\mathcal{D} ; ?A) :: \Gamma ; \Delta \Longrightarrow ?A}\ ?R \qquad \frac{\mathcal{D} :: \Gamma ; u : A \Longrightarrow C\ \mathit{poss}}{?L(u.\ \mathcal{D} ; v{:}?A) :: \Gamma ; v : ?A \Longrightarrow C\ \mathit{poss}}\ ?L$$

Quantifiers

$$\frac{\mathcal{D} :: \Gamma ; \Delta \Longrightarrow [a/x]A}{\forall R(a.\ \mathcal{D} ; \forall x.A) :: \Gamma ; \Delta \Longrightarrow \forall x.A}\ \forall R^a \qquad \frac{\mathcal{D} :: \Gamma ; \Delta, u : [t/x]A \Longrightarrow J}{\forall L(u.\ \mathcal{D}, t ; v : \forall x.A) :: \Gamma ; \Delta, v : \forall x.A \Longrightarrow J}\ \forall L$$

$$\frac{\mathcal{D} :: \Gamma ; \Delta \Longrightarrow [t/x]A}{\exists R(\mathcal{D}, t ; \exists x.A) :: \Gamma ; \Delta \Longrightarrow \exists x.A}\ \exists R \qquad \frac{\mathcal{D} :: \Gamma ; \Delta, u : [a/x]A \Longrightarrow J}{\exists L(a.\ u.\ \mathcal{D} ; v : \exists x.A) :: \Gamma ; \Delta, v : \exists x.A \Longrightarrow J}\ \exists L^a$$

Definition 2.16.

1. *The free labels in a derivation $\mathcal{D}$, written* $\mathrm{fl}(\mathcal{D})$*, are as follows:*

$$\mathrm{fl}(\mathrm{init}(u{:}p(\vec{t}))) = \{u\} \qquad \mathrm{fl}(\mathrm{poss}(\mathcal{D})) = \mathrm{fl}(\mathcal{D})$$

$$\mathrm{fl}(\mathrm{copy}(u.\ \mathcal{D} ; v{:}A)) = \mathrm{fl}(\mathcal{D}) \setminus \{u\}$$

$$\mathrm{fl}(\otimes R(\mathcal{D}_1, \mathcal{D}_2 ; A \otimes B)) = \mathrm{fl}(\mathcal{D}_1) \cup \mathrm{fl}(\mathcal{D}_2) \qquad \mathrm{fl}(\mathbf{1}R) = \emptyset$$

$$\mathrm{fl}(\otimes L(u.\ v.\ \mathcal{D} ; w{:}A \otimes B)) = \mathrm{fl}(\mathcal{D}) \setminus \{u, v\} \cup \{w\} \quad \mathrm{fl}(\mathbf{1}L(\mathcal{D} ; u{:}\mathbf{1})) = \mathrm{fl}(\mathcal{D}) \cup \{u\}$$

$$\mathrm{fl}(\multimap R(u.\ \mathcal{D} ; A \multimap B)) = \mathrm{fl}(\mathcal{D}) \setminus \{u\}$$

$$\mathrm{fl}(\multimap L(\mathcal{D}, u.\ \mathcal{D}' ; v{:}A \multimap B)) = \mathrm{fl}(\mathcal{D}) \cup \mathrm{fl}(\mathcal{D}') \setminus \{u\} \cup \{v\}$$

$$\mathrm{fl}(\&R(\mathcal{D}_1, \mathcal{D}_2 ; A \,\&\, B)) = \mathrm{fl}(\mathcal{D}_1) \cup \mathrm{fl}(\mathcal{D}_2) \qquad \mathrm{fl}(\top R) = \emptyset$$

$$\mathrm{fl}(\&L_i(u.\ \mathcal{D}\,;v\!:\!A_1 \,\&\, A_2)) = \mathrm{fl}(\mathcal{D}) \setminus \{u\} \cup \{v\}$$

$$\mathrm{fl}(\oplus R_i(\mathcal{D}\,;A_1 \oplus A_2)) = \mathrm{fl}(\mathcal{D}) \qquad \mathrm{fl}(\mathbf{0}L(u\!:\!\mathbf{0})) = \{u\}$$

$$\mathrm{fl}(\mathbf{0}L(u.\ \mathcal{D}, v.\ \mathcal{D}'\,;w\!:\!A \oplus B)) = \mathrm{fl}(\mathcal{D}) \setminus \{u\} \cup \mathrm{fl}(\mathcal{D}') \setminus \{v\} \cup \{w\}$$

$$\mathrm{fl}(!R(\mathcal{D}\,;!\,A)) = \mathrm{fl}(\mathcal{D}) \qquad \mathrm{fl}(!L(u.\ \mathcal{D}\,;v\!:\!!A)) = \mathrm{fl}(\mathcal{D}) \setminus \{u\} \cup \{v\}$$

$$\mathrm{fl}(?R(\mathcal{D}\,;?\,A)) = \mathrm{fl}(\mathcal{D}) \qquad \mathrm{fl}(?L(u.\ \mathcal{D}\,;v\!:\!?A)) = \mathrm{fl}(\mathcal{D}) \setminus \{u\} \cup \{v\}$$

$$\mathrm{fl}(\forall R(a.\ \mathcal{D}\,;\forall x.A)) = \mathrm{fl}(\mathcal{D}) \qquad \mathrm{fl}(\forall L(u.\ \mathcal{D}, t\,;v\!:\!\forall x.A) = \mathrm{fl}(\mathcal{D}) \setminus \{u\} \cup \{v\}$$

$$\mathrm{fl}(\exists R(\mathcal{D}, t\,;\exists x.A)) = \mathrm{fl}(\mathcal{D}) \qquad \mathrm{fl}(\exists L(a.\ u.\ \mathcal{D}\,;v\!:\!\exists x.A) = \mathrm{fl}(\mathcal{D}) \setminus \{u\} \cup \{v\}$$

Theorem 2.17 (Structural properties)**.**

1. *If $\mathcal{D} :: \Gamma\,;\Delta \Longrightarrow J$, then $\mathcal{D} :: \Gamma, \Gamma'\,;\Delta \Longrightarrow J$. (weakening)*
2. *If $\mathcal{D} :: \Gamma, u : A, v : A\,;\Delta \Longrightarrow J$, then $[u/v]\mathcal{D} :: \Gamma, u : A\,;\Delta \Longrightarrow J$. (contraction)*

Proof. By induction on the structure of $\mathcal{D}$ in each case. □

We shall present the identity principle using a computational procedure that constructs the identity derivation, and then prove that the constructed derivation is a valid derivation for the identity.

Definition 2.18 (Identity computation)**.** *We define the operation "id" from labelled propositions to sequent derivations, hereditarily, using the following equations.*

$$\mathrm{id}(u\!:\!p(\vec{t})) = \mathrm{init}(u\!:\!p(\vec{t}))$$

$$\begin{aligned}
\mathrm{id}(u\!:\!A \otimes B) &= \otimes L(v.\ w.\ \otimes R(\mathrm{id}(v\!:\!A), \mathrm{id}(w\!:\!B)\,;A \otimes B)\,;u\!:\!A \otimes B)\\
\mathrm{id}(u\!:\!\mathbf{1}) &= \mathbf{1}L(\mathbf{1}R\,;u\!:\!\mathbf{1})\\
\mathrm{id}(u\!:\!A \multimap B) &= \multimap R(v.\ \multimap L(\mathrm{id}(v\!:\!A), w.\ \mathrm{id}(w\!:\!B)\,;u\!:\!A \multimap B)\,;A \multimap B)
\end{aligned}$$

$$\begin{aligned}
\mathrm{id}(u\!:\!A \,\&\, B) &= \&R(\&L_1(v_1.\ \mathrm{id}(v_1\!:\!A)\,;u\!:\!A \,\&\, B), \&L_2(v_2.\ \mathrm{id}(v_2\!:\!B)\,;u\!:\!A \,\&\, B)\,;A \,\&\, B)\\
\mathrm{id}(u\!:\!\top) &= \top R\\
\mathrm{id}(u\!:\!A \oplus B) &= \oplus L(v.\ \oplus R_1(\mathrm{id}(v\!:\!A)\,;A \oplus B), w.\ \oplus R_2(\mathrm{id}(w\!:\!B)\,;A \oplus B)\,;u\!:\!A \oplus B)\\
\mathrm{id}(u\!:\!\mathbf{0}) &= \mathbf{0}L(u\!:\!\mathbf{0})
\end{aligned}$$

$$\begin{aligned}
\mathrm{id}(u\!:\!!A) &= !L(v.\ !R(\mathrm{copy}(w.\ \mathrm{id}(w : A)\,;v\!:\!A)\,;!A)\,;u\!:\!!A)\\
\mathrm{id}(u\!:\!?A) &= ?R(?L(v.\ \mathrm{poss}(\mathrm{id}(v\!:\!A))\,;u\!:\!?A)\,;?A)
\end{aligned}$$

$$\mathrm{id}(u{:}\forall x.A) = \forall R(a.\ \forall L(v.\ \mathrm{id}(v{:}[a/x]A)\,;u{:}\forall x.A)\,;\forall x.A)$$
$$\mathrm{id}(u{:}\exists x.A) = \exists L(a.\ v.\ \exists R(\mathrm{id}(v{:}[a/x]A), a\,;\exists x.A)\,;u{:}\exists x.A)$$

Theorem 2.19 (Identity principle)**.**
For any proposition A and context Γ, we have $\mathrm{id}(u{:}A) :: \Gamma\,; u{:}A \Longrightarrow A$.

Proof. Induction on the structure of A, using definition 2.18 as appropriate. The following is a representative example.

$$\dfrac{\dfrac{\dfrac{}{\mathrm{id}(v:A) :: \Gamma\,; v:A \Longrightarrow A}\ \text{i.h.} \quad \dfrac{}{\mathrm{id}(w:B) :: \Gamma\,; w:B \Longrightarrow B}\ \text{i.h.}}{\otimes R(\mathrm{id}(v:A), \mathrm{id}(w:B)\,; A\otimes B) :: \Gamma\,; v:A, w:B \Longrightarrow A\otimes B}\ \otimes R}{\otimes L(v.\ w.\ \otimes R(\mathrm{id}(v:A), \mathrm{id}(w:B)\,; A\otimes B)\,; u:A\otimes B) :: \Gamma\,; u{:}A\otimes B \Longrightarrow A\otimes B}\ \otimes L \qquad \square$$

Cut elimination will be presented as a computation on derivations.

Definition 2.20 (Cut elimination as computation)**.** *We define the binary operations $+$, $+_!$ and $+_?$ on derivations such that*

1. *if $\mathcal{D} :: \Gamma\,; \Delta \Longrightarrow A$ goal and $\mathcal{E} :: \Gamma\,; \Delta', u{:}A \Longrightarrow J$, then $\mathcal{D} +_{u:A} \mathcal{E}$ is defined.*
2. *if $\mathcal{D} :: \Gamma\,; \cdot \Longrightarrow A$ goal and $\mathcal{E} :: \Gamma, u:A\,; \Delta \Longrightarrow J$, then $\mathcal{D} +_{!u:A} \mathcal{E}$ is defined.*
3. *if $\mathcal{D} :: \Gamma\,; \Delta \Longrightarrow A$ poss and $\mathcal{E} :: \Gamma\,; u:A \Longrightarrow C$ poss, then $\mathcal{D} +_{?u:A} \mathcal{E}$ is defined.*

These three operations are defined in a mutually recursive fashion as follows. We use the convention that primed labels are fresh, i.e., not occurring freely in any of the subderivations.

a. ***Initial cases***

$$\mathrm{init}(u : p(\vec{t})) +_{v:p(\vec{t})} \mathcal{E} = [u/v]\mathcal{E}$$
$$\mathcal{D} +_{u:p(\vec{t})} \mathrm{init}(u : p(\vec{t})) = \mathcal{D}$$

b. ***Principal cases***

$$\otimes R(\mathcal{D}_1, \mathcal{D}_2\,; A\otimes B) +_{w:A\otimes B} \otimes L(u.\ v.\ \mathcal{E}\,; w{:}A\otimes B) = \mathcal{D}_1 +_{u':A} (\mathcal{D}_2 +_{v':B} [u'/u, v'/v]\mathcal{E})$$
$$\mathbf{1}R +_{u:\mathbf{1}} \mathbf{1}L(\mathcal{E}\,; u{:}\mathbf{1}) = \mathcal{E}$$
$$\multimap R(u.\ \mathcal{D}\,; A\multimap B) +_{w:A\multimap B} \multimap L(\mathcal{E}_1, v.\ \mathcal{E}_2\,; w{:}A\multimap B) = (\mathcal{E}_1 +_{u':A} [u'/u]\mathcal{D}) +_{v':B} [v'/v]\mathcal{E}_2$$
$$\& R(\mathcal{D}_1, \mathcal{D}_2\,; A_1 \,\&\, A_2) +_{v:A_1\&A_2} \& L_i(u.\ \mathcal{E}\,; v{:}A_1 \,\&\, A_2) = \mathcal{D}_i +_{u':A_i} [u'/u]\mathcal{E}$$
$$\oplus R_i(\mathcal{D}\,; A_1\oplus A_2) +_{w:A\oplus B} \oplus L(u_1.\ \mathcal{E}_1, u_2.\ \mathcal{E}_2\,; w : A\oplus B) = \mathcal{D} +_{u_i':A_i} [u_i'/u_i]\mathcal{E}_i$$
$$!R(\mathcal{D}\,; !A) +_{v:!A} !L(u.\ \mathcal{E}\,; v{:}!A) = \mathcal{D} +_{!u':A} [u'/u]\mathcal{E}$$

$$?R(\mathcal{D}\,;?A) +_{v:?A} ?L(u.\,\mathcal{E}\,;v : ?A) = \mathcal{D} +_{?u':A} [u'/u]\mathcal{E}$$

$$\forall R(a.\,\mathcal{D}\,;\forall x.A) +_{v:\forall x.A} \forall L(u.\,\mathcal{E},t\,;v : \forall x.A) = [t/a]\mathcal{D} +_{u':[t/x]A} [u'/u]\mathcal{E}$$

$$\exists R(\mathcal{D},t\,;\exists x.A) +_{v:\exists x.A} \exists L(a.\,u.\,\mathcal{E}\,;v : \exists x.A) = \mathcal{D} +_{u':[t/x]A} [t/a][u'/u]\mathcal{E}$$

c. ***Left-commutative cases*** *(here $+\!\!\!+$ stands for either $+$ or $+_?$)*

$$\otimes L(u.\,v.\,\mathcal{D}\,;w{:}A\otimes B) +\!\!\!+_{z:C} \mathcal{E} = \otimes L(u'.\,v'.\,[u'/u,v'/v]\mathcal{D} +\!\!\!+_{z:C} \mathcal{E}\,;w{:}A\otimes B)$$

$$\mathbf{1}L(\mathcal{D}\,;u{:}\mathbf{1}) +\!\!\!+_{v:C} \mathcal{E} = \mathbf{1}L(\mathcal{D} +\!\!\!+_{v:C} \mathcal{E}\,;u{:}\mathbf{1})$$

$$\multimap L(\mathcal{D},u.\,\mathcal{D}'\,;v{:}A\multimap B) +\!\!\!+_{w:C} \mathcal{E} = \multimap L(\mathcal{D},u'.\,[u'/u]\mathcal{D}' +\!\!\!+_{w:C} \mathcal{E}\,;v{:}A\multimap B)$$

$$\& L_i(u.\,\mathcal{D}\,;v{:}A\,\&\,B) +\!\!\!+_{w:C} \mathcal{E} = \& L_i(u'.\,[u'/u]\mathcal{D} +\!\!\!+_{w:C} \mathcal{E}\,;v{:}A\,\&\,B)$$

$$\oplus L(u.\,\mathcal{D},v.\,\mathcal{D}'\,;w : A\oplus B) +\!\!\!+_{z:C} \mathcal{E} = \oplus L(u'.\,[u'/u]\mathcal{D} +\!\!\!+_{z:C} \mathcal{E},v'.\,[v'/v]\mathcal{D}' +\!\!\!+_{z:C} \mathcal{E}\,;w{:}A\oplus B)$$

$$\mathbf{0}L(u{:}\mathbf{0}) +\!\!\!+_{v:C} \mathcal{E} = \mathbf{0}L(u{:}\mathbf{0})$$

$$!L(u.\,\mathcal{D}\,;v{:}!A) +\!\!\!+_{v:C} \mathcal{E} = !L(u'.\,[u'/u]\Delta +\!\!\!+_{v:C} \mathcal{E}\,;v{:}!A)$$

$$?L(u.\,\mathcal{D}\,;v{:}?A) +_{v:C} \mathcal{E} = ?L(u'.\,[u'/u]\Delta +\!\!\!+_{v:C} \mathcal{E}\,;v{:}?A)$$

$$\forall L(u.\,\mathcal{D},t\,;v{:}\forall x.A) +\!\!\!+_{w:C} \mathcal{E} = \forall L(u'.\,[u'/u]\mathcal{D} +\!\!\!+_{w:C} \mathcal{E},t\,;v{:}\forall x.A)$$

$$\exists L(a.\,u.\,\mathcal{D}\,;v{:}\exists x.A) +\!\!\!+_{w:C} \mathcal{E} = \exists L(a.\,u'.\,[u'/u]\mathcal{D} +\!\!\!+_{w:C} \mathcal{E}\,;v{:}\exists x.A)$$

$$\mathrm{copy}(u.\,\mathcal{D}\,;v\,;A) +\!\!\!+_{w:C} \mathcal{E} = \mathrm{copy}(u'.\,[u'/u]\mathcal{D} +\!\!\!+_{w:C} \mathcal{E},v\,;A)$$

d. ***Right-commutative cases*** *(here $+\!\!\!+$ stands for either $+$ or $+_!$)*

$$\mathcal{D} +\!\!\!+_{u:C} \otimes R(\mathcal{E},\mathcal{E}'\,;A\otimes B) = \begin{cases} \otimes R(\mathcal{D} +\!\!\!+_{u:C} \mathcal{E},\mathcal{E}'\,;A\otimes B) & \textit{if } u \in \mathrm{fl}(\mathcal{E}), \textit{ or} \\ \otimes R(\mathcal{E},\mathcal{D} +\!\!\!+_{u:C} \mathcal{E}'\,;A\otimes B) & \textit{if } u \in \mathrm{fl}(\mathcal{E}') \end{cases}$$

$$\mathcal{D} +\!\!\!+_{u:C} \otimes L(v_1.\,v_2.\,\mathcal{E}\,;w{:}A\otimes B) = \otimes L(v_1'.\,v_2'.\,\mathcal{D} +\!\!\!+_{u:C} [v_1'/v_1,v_2'/v_2]\mathcal{E}\,;w{:}A\otimes B)$$

$$\mathcal{D} +\!\!\!+_{u:C} \multimap R(v.\,\mathcal{E}\,;A\multimap B) = \multimap R(v'.\,\mathcal{D} +\!\!\!+_{u:C} [v'/v]\mathcal{E}\,;A\multimap B)$$

$$\mathcal{D} +\!\!\!+_{u:C} \multimap L(\mathcal{E},v.\,\mathcal{E}'\,;w : A\multimap B) = \begin{cases} \multimap L(\mathcal{D} +\!\!\!+_{u:C} \mathcal{E},v.\,\mathcal{E}'\,;w : A\multimap B) & \textit{if } u \in \mathrm{fl}(\mathcal{E}), \textit{ or} \\ \multimap L(\mathcal{E},v'.\,\mathcal{D},+\!\!\!+_{u:C}[v'/v]\mathcal{E}'\,;w : A\multimap B) & \textit{if } u \in \mathrm{fl}(\mathcal{E}') \end{cases}$$

$$\mathcal{D} +\!\!\!+_{u:C} \& R(\mathcal{E},\mathcal{E}'\,;A\,\&\,B) = \& R(\mathcal{D} +\!\!\!+_{u:C} \mathcal{E},\mathcal{D} +\!\!\!+_{u:C} \mathcal{E}'\,;A\,\&\,B)$$

$$\mathcal{D} +\!\!\!+_{u:C} \& L_i(v.\,\mathcal{E}\,;w{:}A\,\&\,B) = \& L_i(v'.\,\mathcal{D} +\!\!\!+_{u:C} [v'/v]\mathcal{E}\,;w{:}A\,\&\,B)$$

$$\mathcal{D} +\!\!\!+_{u:C} \top R = \top R$$

$$\mathcal{D} +\!\!\!+_{u:C} \oplus R_i(\mathcal{E}\,;A\oplus B) = \oplus R_i(\mathcal{D} +\!\!\!+_{u:C} \mathcal{E}\,;A\oplus B)$$

$$\mathcal{D} +\!\!\!+_{u:C} \oplus L(v.\,\mathcal{E},w.\,\mathcal{E}'\,;z{:}A\oplus B) = \oplus L(v'.\,\mathcal{D} +\!\!\!+_{u:C} [v'/v]\mathcal{E},w'.\,\mathcal{D} +_{u:C} [w'/w]\mathcal{E}'\,;z{:}A\oplus B)$$

$$\mathcal{D} +\!\!\!+_{u:C} \mathbf{0}L(v{:}\mathbf{0}) = \mathbf{0}L(v{:}\mathbf{0})$$

$$\mathcal{D} +_{!u:C} !R(\mathcal{E}\,;!A) = !R(\mathcal{D} +_{!u:C} \mathcal{E}\,;!A)$$

$$\begin{aligned}
\mathcal{D} +_{!u:C} {!L}(v.\, \mathcal{E}\,;\, w\!:\!{!A}) &= {!L}(v'.\, \mathcal{D} +_{u:C} [v'/v]\mathcal{E}\,;\, w\!:\!{!A})\\
\mathcal{D} +\!\!+_{u:C} {?R}(\mathcal{E}\,;\, ?A) &= {?R}(\mathcal{D} +\!\!+_{u:C} \mathcal{E}\,;\, ?A)\\
\mathcal{D} +_{!u:C} {?L}(v.\, \mathcal{E}\,;\, w\!:\!?A) &= {?L}(v'.\, \mathcal{D} +_{u:C} [v'/v]\mathcal{E}\,;\, w\!:\!?A)\\
\mathcal{D} +\!\!+_{u:C} \forall R(a.\, \mathcal{E}\,;\, \forall x.A) &= \forall R(a.\, \mathcal{D} +\!\!+_{u:C} \mathcal{E}\,;\, \forall x.A)\\
\mathcal{D} +\!\!+_{u:C} \forall L(v.\, \mathcal{E}, t\,;\, w\!:\!\forall x.A) &= \forall L(v'.\, \mathcal{D} +\!\!+_{u:C} [v'/v]\mathcal{E}, t\,;\, w\!:\!\forall x.A)\\
\mathcal{D} +\!\!+_{u:C} \exists R(\mathcal{E}, t\,;\, \exists x.A) &= \exists R(\mathcal{D} +\!\!+_{u:C} \mathcal{E}, t\,;\, \exists x.A)\\
\mathcal{D} +\!\!+_{u:C} \exists L(a.\, v.\, \mathcal{E}\,;\, w\!:\!\exists x.A) &= \exists L(a.\, v'.\, \mathcal{D} +\!\!+_{u:C} [v'/v]\mathcal{E}\,;\, w\!:\!\exists x.A)\\
\mathcal{D} +\!\!+_{u:C} \mathrm{copy}(v.\, \mathcal{E}\,;\, w\!:\!A) &= \mathrm{copy}(v'.\, \mathcal{D} +\!\!+_{u:C} [v'/v]\mathcal{E}\,;\, w\!:\!A)\\
\mathcal{D} +\!\!+_{u:C} \mathrm{poss}(\mathcal{E}) &= \mathrm{poss}(\mathcal{D} +\!\!+_{u:C} \mathcal{E})
\end{aligned}$$

For the $\otimes R$ and $\multimap L$ cases only one of the two possibilities will be defined.

e. ***Structural cases***

$$\begin{aligned}
\mathcal{D} +_{!u:A} \mathrm{copy}(u.\, \mathcal{E}\,;\, v\!:\!A) &= \mathcal{D} +_{u':A} (\mathcal{D} +_{!v:A} [u'/u]\mathcal{E})\\
\mathrm{poss}(\mathcal{D}) +_{?u:A} \mathcal{E} &= \mathcal{D} +_{u:A} \mathcal{E}
\end{aligned}$$

We then prove that these operations correspond exactly to the three cases of cut.

Theorem 2.21 (Cut elimination)**.**

(1) If $\mathcal{D} :: \Gamma\,;\Delta \Longrightarrow A$ & $\mathcal{E} :: \Gamma\,;\Delta', u : A \Longrightarrow J$, then $\mathcal{D} +_{u:A} \mathcal{E} :: \Gamma\,;\Delta, \Delta' \Longrightarrow J$.
(2) If $\mathcal{D} :: \Gamma\,;\cdot \Longrightarrow A$ & $\mathcal{E} :: \Gamma, u : A\,;\Delta \Longrightarrow J$, then $\mathcal{D} +_{!u:A} \mathcal{E} :: \Gamma\,;\Delta \Longrightarrow J$.
(3) If $\mathcal{D} :: \Gamma\,;\Delta \Longrightarrow A$ poss & $\mathcal{E} :: \Gamma\,;u : A \Longrightarrow C$ poss, then $\mathcal{D}+_{?u:A} :: \Gamma\,;\Delta \Longrightarrow C$ poss.

Proof. The proof will use a lexicographic induction on the structure of $\mathcal{D}$ and $\mathcal{E}$. We shall assume that the inductive hypothesis can be used whenever:

1. the cut formula is strictly smaller; or
2. the cut formula remains the same, but the inductive hypothesis is used for cut kind (3) in proofs of cut kind (2) and cut kind (1), or for cut kind (2) in proofs of cut kind (1); or
3. the cut formula and $\mathcal{E}$ remain the same, and $\mathcal{D}$ is strictly smaller; or
4. the cut formula and $\mathcal{D}$ remain the same, and $\mathcal{E}$ is strictly smaller.

Initial cuts Here, either $\mathcal{D}$ or $\mathcal{E}$ ends with "init".

Case. $\mathcal{D} = \text{init}(u{:}p(\vec{t})) :: \Gamma\,;u{:}p(\vec{t}) \Longrightarrow p(\vec{t})$ and $\mathcal{E} :: \Gamma\,;\Delta, v : p(\vec{t}) \Longrightarrow J$.

In this case, $\mathcal{F} = [u/v]\mathcal{E} :: \Gamma\,;\Delta, u : p(\vec{t}) \Longrightarrow p(\vec{t})$.

Cbse. $\mathcal{D} :: \Gamma\,;\Delta \Longrightarrow p(\vec{t})$ and $\mathcal{E} = \text{init}(v{:}p(\vec{t})) :: \Gamma\,;v : p(\vec{t}) \Longrightarrow p(\vec{t})$. In this case, $\mathcal{F} = \mathcal{D}$.

Principal cuts In these cases, the cut formula was last inferred by a right rule in $\mathcal{D}$ and a left rule in $\mathcal{E}$. All of these cases have been sketched before. We shall give only one representative case:

$$\frac{\mathcal{D}_1 :: \Gamma\,;\Delta \Longrightarrow A \quad \mathcal{D}_2 :: \Gamma\,;\Delta' \Longrightarrow B}{\mathcal{D} = \otimes R(\mathcal{D}_1, \mathcal{D}_2\,;A \otimes B) :: \Gamma\,;\Delta, \Delta' \Longrightarrow A \otimes B}\,\otimes R$$

$$\frac{\mathcal{E}' :: \Gamma\,;\Delta', u{:}A, v{:}B \Longrightarrow J}{\mathcal{E} = \otimes L(u.\,v.\,\mathcal{E}'\,;w{:}A \otimes B) :: \Gamma\,;\Delta', w{:}A \otimes B \Longrightarrow J}\,\otimes L$$

$\mathcal{D}_1 +_{u':A} [u'/u, v'/v]\mathcal{E}' :: \Gamma\,;\Delta, \Delta'', v'{:}B \Longrightarrow J$ — i.h. (smaller cut formula)

$\mathcal{D}_2 +_{v':B} (\mathcal{D}_1 +_{u':A} [u'/u, v'/v]\mathcal{E}') :: \Gamma\,;\Delta, \Delta', \Delta'' \Longrightarrow J$ — i.h. (smaller cut formula)

Truth cuts All cuts of kind (2) are treated as right-commutative cuts *except* for the case where the last rule in $\mathcal{E}$ is "copy". In this case, $\mathcal{D} :: \Gamma\,;\cdot \Longrightarrow A$ and

$$\frac{\mathcal{E}' :: \Gamma, u : A\,;\Delta, v : A \Longrightarrow J}{\mathcal{E} = \text{copy}(v.\,\mathcal{E}'\,;u{:}A) :: \Gamma, u : A\,;\Delta \Longrightarrow J}\,\text{copy.}$$

$\mathcal{D} +_{!u:A} [v'/v]\mathcal{E}' :: \Gamma\,;\Delta, v' : A \Longrightarrow J$ — i.h. ($\mathcal{E}'$ smaller than $\mathcal{E}$)

$\mathcal{D} +_{v':A} (\mathcal{D} +_{!u:A} [v'/v]\mathcal{E}') :: \Gamma\,;\Delta \Longrightarrow J$ — i.h. ((2) used to justify (1))

Possibility cuts All cuts of kind (3) are treated as left-commutative cuts *except* for the case where the last rule in $\mathcal{D}$ is "poss". In this case, $\mathcal{E} :: \Gamma\,;u{:}A \Longrightarrow C\ \textit{poss}$ and

$$\frac{\mathcal{D}' :: \Gamma\,;\Delta \Longrightarrow A\ \textit{goal}}{\mathcal{D} = \text{poss}(\mathcal{D}') :: \Gamma\,;\Delta \Longrightarrow A\ \textit{poss}}\,\text{poss}$$

$\mathcal{D}' +_{?u:A} \mathcal{E} :: \Gamma\,;\Delta \Longrightarrow C\ \textit{poss}$ — i.h. ($\mathcal{D}'$ smaller than $\mathcal{D}$)

Left commutative cuts The cut formula is a side formula in the last inference used in $\mathcal{D}$. In these cases we appeal to the induction hypotheses with the same cut formula, but a smaller left derivation. The following is a representative case.

$$\frac{\mathcal{D}' :: \Gamma\,;\Delta, u{:}B \Longrightarrow A}{\mathcal{D} = \&L_1(u.\,\mathcal{D}'\,;v{:}A \,\&\, B) :: \Gamma\,;\Delta, v{:}B \,\&\, C \Longrightarrow A}\,\&L_1 \qquad \mathcal{E} :: \Gamma\,;\Delta', w{:}A \Longrightarrow J$$

$$\mathcal{D}' +_{w:A} [u'/u]\mathcal{E} :: \Gamma\,;\Delta, u' : B \Longrightarrow J \qquad \text{i.h. } (\mathcal{D}' \text{ smaller than } \mathcal{D})$$
$$\&L_1(u'.\,\mathcal{D}' +_{w:A} [u'/u]\mathcal{E}\,; w{:}B \,\&\, C) :: \Gamma\,;\Delta, w{:}B \,\&\, C \Longrightarrow J \qquad \&L_1$$

Right commutative cuts The cut formula is a side formula in the last inference used in $\mathcal{E}$. In these cases we appeal to the induction hypotheses with the same cut formula, but a smaller right derivation. The following is a representative case.

$$\mathcal{D} :: \Gamma\,;\Delta \Longrightarrow A \qquad \frac{\mathcal{E}' :: \Gamma\,;\Delta', u : A \Longrightarrow B}{\mathcal{E} = \oplus R_1(\mathcal{E}'\,; B \oplus C) :: \Gamma\,;\Delta', u : A \Longrightarrow B \oplus C}\,\oplus R_1$$

$$\mathcal{D} +_{u:A} \mathcal{E}' :: \Gamma\,;\Delta, \Delta' \Longrightarrow B \qquad \text{i.h. } (\mathcal{E}' \text{ smaller than } \mathcal{E})$$
$$\oplus R_1(\mathcal{D} +_{u:A} \mathcal{E}'\,; B \oplus C) :: \Gamma\,;\Delta, \Delta' \Longrightarrow B \oplus C \qquad \oplus R_1$$

This completes the inventory of all cuts. □

Comparing this proof of cut elimination with other proofs of cut-admissibility or cut-elimination in the literature, it is worth remarking that a nested structural induction suffices. No additional restrictions on the cut rules or induction measures are required. Similar structural and constructive proofs for cut-admissibility have been demonstrated for classical linear logic [93], classical and intuitionistic uniform sequent calculi [86, 87] intuitionistic contraction-free logic [88] and ordered logic [96].

2.2 Natural deduction and proofs

While the syntax of derivations introduced in the previous section is fine for proof objects, it is conceptually and presentationally awkward. A more natural presentation of logic in Martin-Löf's philosophy of logic use *natural deduction* as the foundation. In this section, we present a translation of sequent derivations into natural deduction proofs. In fact, the natural deduction proofs that will be generated from our cut-free sequent setting will be in β-normal and η-long form.

In natural deduction, we discard the duality of resources and goals and have just a single judgement form, *A res*. All rules always operate on the right hand side of the *linear hypothetical judgement*. Like before, we have judgemental rules for the structural properties of hypothetical judgements, and logical rules for the various connectives. We also adopt the judgements $A\ poss$ from the sequent calculus setting in an analogous form.

Proofs for the judgemental forms $A\ res$ and $A\ poss$ will be called proof *terms* (written $M, N, \ldots$) and *expressions* (written $E, F, \ldots$), respectively. The hypothetical natural deduction judgements are of the form $\Gamma\, ; \Delta \vdash M{:}A\ res$ or $\Gamma\, ; \Delta \vdash E \div A\ poss$. Like before, Γ contains the unrestricted hypotheses, Δ the linear hypotheses, and J the conclusion form (either $C\ res$ or $C\ poss$). For convenience and brevity, we omit the judgemental labels when understood. The right hand forms $M{:}A\ res$ and $E \div A\ poss$ are represented schematically as $P * J$, with P standing for the witness (M or E), and J for the judgement ($A\ res$ or $A\ poss$).

Judgemental rules The first judgemental rules define the use of hypotheses. We have two rules, one for each kind of hypothesis.

$$\frac{}{\Gamma\, ; u{:}A \vdash u{:}A}\ \text{hyp} \qquad \frac{}{\Gamma, u : A\, ; \cdot \vdash u{:}A}\ \text{hyp!} \qquad \frac{\Gamma\, ; \Delta \vdash M{:}A\ res}{\Gamma\, ; \Delta \vdash M \div A\ poss}\ \text{poss}$$

The proof for either "hyp" rule is the label u of the resource that matches the right hand side. For the poss rule, the same proof term counts as a proof expression; this is sometimes called a *silent coercion* in the literature.

Dually, we may substitute proofs for uses of a resource, which we define as a *substitution principle*. This theorem relies on two syntactic substitution operations on proof terms and expressions, written $[M/u]P$ and $\langle E/u \rangle\, F$ respectively, the full definition of which is delayed until the details of proof terms and expressions are presented (see defn. 2.23).

Principle 2.22 (Substitution)**.**

1. *If* $\Gamma\, ; \Delta \vdash M{:}A$ *and* $\Gamma\, ; \Delta', u{:}A \vdash P * J$*, then* $\Gamma\, ; \Delta, \Delta' \vdash [M/u]P * J$.
2. *If* $\Gamma\, ; \cdot \vdash M{:}A$ *and* $\Gamma, u : A\, ; \Delta' \vdash P * J$*, then* $\Gamma\, ; \Delta \vdash [M/u]P * J$.
3. *If* $\Gamma\, ; \Delta \vdash E{:}A\ poss$ *and* $\Gamma\, ; u{:}A \vdash F \div C\ poss$*, then* $\Gamma\, ; \Delta, \Delta' \vdash \langle E/u \rangle\, F \div C\ poss$.

Like with cut in the sequent calculus, we do not realise these substitution principles as inference rules but intend them as structural properties of the logic maintained by all other inference rules. We can prove the substitution principles by induction over the structure of derivations. Meanwhile, the substitution principles can be used to show local soundness and completeness of the inference rules characterising the connectives.

Logical rules In the judgemental philosophy, the meaning of a propositional connective is defined by its *introduction rule(s)*. Once a goal such as $A \otimes B$ is established, the *elimination rule(s)* for the topmost connective, $\otimes$ in this case, describe the means of decomposing the goal into simpler goals, or deriving resources from that goal for establishing other goals. For each connective, the introduction and elimination rules must together satisfy two consistency criteria.

Local soundness criterion: given sufficient evidence for the premisses of an inference rule, we must be able to find sufficient evidence for the conclusion of the rule. This criterion manifests a check on the strength of elimination rules – they must not derive evidence not already implied by the premisses. The usual prescription is to define this criterion by means of a *local reduction*, written $\Longrightarrow_R$, which shows how to transform a proof containing an introduction of a connective followed immediately by its elimination into a proof without this detour.

Local completeness criterion: by eliminating a given propositional connective, we must obtain enough evidence to reconstitute the connective. Again, the usual approach is to define this criterion by means of a *local expansion*, written $\Longrightarrow_E$, which shows how to transform a proof of a given proposition into one that introduces its main connective.

I will illustrate these criteria with one example, that of the multiplicative conjunction, $\otimes$. Its introduction and elimination rules are follows.

$$\frac{\Gamma\,;\Delta_1 \vdash M\!:\!A \quad \Gamma\,;\Delta_2 \vdash N\!:\!B}{\Gamma\,;\Delta_1,\Delta_2 \vdash (M \otimes N)\!:\!A \otimes B}\ \otimes I$$

$$\frac{\Gamma\,;\Delta_1 \vdash M\!:\!A \otimes B \quad \Gamma\,;\Delta_2, u\!:\!A, v\!:\!B \vdash P * J}{\Gamma\,;\Delta_1,\Delta_2 \vdash (\mathbf{let}\, u \otimes v = M\, \mathbf{in}\, P) * J}\ \otimes E$$

Local soundness is the following reduction:

$$\frac{\dfrac{\Gamma\,;\Delta_1 \vdash M\!:\!A \quad \Gamma\,;\Delta_2 \vdash N\!:\!B}{\Gamma\,;\Delta_1,\Delta_2 \vdash (M \otimes N)\!:\!A \otimes B}\ \otimes I \quad \Gamma\,;\Delta_3, u\!:\!A, v\!:\!B \vdash P * J}{\Gamma\,;\Delta_1,\Delta_2,\Delta_3 \vdash (\mathbf{let}\, u \otimes v = M \otimes N\, \mathbf{in}\, P) * J}\ \otimes E$$

$$\Longrightarrow_R \quad \Gamma\,;\Delta_1,\Delta_2,\Delta_3 \vdash [M/u, N/v]P * J.$$

This reduction in the proof-terms is the familiar β-reduction:

$$\mathbf{let}\, u \otimes v = M \otimes N\, \mathbf{in}\, P \quad \Longrightarrow_R \quad [M/u, N/v]P.$$

Conversely, local expansion is the following transformation:

$$\Gamma\,;\Delta \vdash M{:}A \otimes B \quad \Longrightarrow_E$$

$$\dfrac{\Gamma\,;\Delta \vdash M{:}A \otimes B \qquad \dfrac{\dfrac{}{\Gamma\,;u{:}A \vdash u{:}A}\,\mathrm{hyp} \quad \dfrac{}{\Gamma\,;v{:}B \vdash v{:}B}\,\mathrm{hyp}}{\Gamma\,;u{:}A, v{:}B \vdash (u \otimes v){:}A \otimes B}\,{\otimes}I}{\Gamma\,;\Delta \vdash (\mathbf{let}\, u \otimes v = M\, \mathbf{in}\, u \otimes v){:}A \otimes B}\,{\otimes}E$$

For the proof-terms alone, this is again the familiar η-expansion:

$$M{:}A \otimes B \quad \Longrightarrow_E \quad \mathbf{let}\, u \otimes v = M\, \mathbf{in}\, u \otimes v$$

Multiplicative rules The following are the rules for the multiplicative connectives $\otimes$, $\mathbf{1}$ and $\multimap$.

$$\dfrac{\Gamma\,;\Delta_1 \vdash M{:}A \quad \Gamma\,;\Delta_2 \vdash N{:}B}{\Gamma\,;\Delta_1, \Delta_2 \vdash (M \otimes N){:}A \otimes B}\,{\otimes}I \qquad \dfrac{\Gamma\,;\Delta_1 \vdash M{:}A \otimes B \quad \Gamma\,;\Delta_2, u{:}A, v{:}B \vdash P * J}{\Gamma\,;\Delta_1, \Delta_2 \vdash (\mathbf{let}\, u \otimes v = M\, \mathbf{in}\, P) * J}\,{\otimes}E$$

$$\dfrac{}{\Gamma\,;\cdot \vdash *{:}\mathbf{1}}\,\mathbf{1}I \qquad \dfrac{\Gamma\,;\Delta \vdash M{:}\mathbf{1} \quad \Gamma\,;\Delta' \vdash P * J}{\Gamma\,;\Delta, \Delta' \vdash (\mathbf{let} * = M\, \mathbf{in}\, P) * J}\,\mathbf{1}E$$

$$\dfrac{\Gamma\,;\Delta, u{:}A \vdash M{:}B}{\Gamma\,;\Delta \vdash (\lambda u.\, M){:}A \multimap B}\,{\multimap}I \qquad \dfrac{\Gamma\,;\Delta \vdash M{:}A \multimap B \quad \Gamma\,;\Delta' \vdash N{:}A}{\Gamma\,;\Delta, \Delta' \vdash (M\,N){:}B}\,{\multimap}E$$

Local reduction and expansion are as follows:

$$\begin{array}{cc}
\mathbf{let}\, u \otimes v = M \otimes N\, \mathbf{in}\, P \Longrightarrow_R [M/u, N/v]P & M{:}A \otimes B \Longrightarrow_E \mathbf{let}\, u \otimes v = M\, \mathbf{in}\, u \otimes v \\
\mathbf{let} * = *\, \mathbf{in}\, P \Longrightarrow_R P & M{:}\mathbf{1} \Longrightarrow_E \mathbf{let} * = M\, \mathbf{in}\, * \\
(\lambda u.\, M)\, N \Longrightarrow_R [N/u]M & M{:}A \multimap B \Longrightarrow_E \lambda u.\, M\, u
\end{array}$$

Additive rules The rules, local reductions, and local expansions for $\&$, $\top$, $\oplus$ and $\mathbf{0}$ are as follows.

$$\dfrac{\Gamma\,;\Delta \vdash M{:}A \quad \Gamma\,;\Delta \vdash N{:}B}{\Gamma\,;\Delta \vdash (M, N){:}A \,\&\, B}\,\&I \qquad \dfrac{\Gamma\,;\Delta \vdash M{:}A \,\&\, B}{\Gamma\,;\Delta \vdash (\mathbf{fst}\, M){:}A}\,\&E_1 \qquad \dfrac{\Gamma\,;\Delta \vdash M{:}A \,\&\, B}{\Gamma\,;\Delta \vdash (\mathbf{snd}\, M){:}B}\,\&E_2$$

$$\dfrac{\Gamma\,;\Delta \vdash M{:}A}{\Gamma\,;\Delta \vdash (\mathbf{inl}\, M){:}A \oplus B}\,{\oplus}I_1 \qquad \dfrac{\Gamma\,;\Delta \vdash M{:}B}{\Gamma\,;\Delta \vdash (\mathbf{inr}\, M){:}A \oplus B}\,{\oplus}I_2$$

$$\dfrac{\Gamma\,;\Delta \vdash M{:}A \oplus B \quad \Gamma\,;\Delta', u{:}A \vdash P * J \quad \Gamma\,;\Delta', v{:}B \vdash Q * J}{\Gamma\,;\Delta, \Delta' \vdash (\mathbf{case}\, M\, \mathbf{of}\, \mathbf{inl}\, u \Rightarrow P \mid \mathbf{inr}\, v \Rightarrow Q) * J}\,{\oplus}E$$

$$\frac{}{\Gamma\,;\Delta \vdash ():\top}\ \top I \qquad \frac{\Gamma\,;\Delta \vdash M:\mathbf{0}}{\Gamma\,;\Delta \vdash (\mathbf{abort}\,M) * J}\ \mathbf{0}E$$

$$\mathbf{fst}(M,N) \Longrightarrow_R M \qquad \mathbf{snd}(M,N) \Longrightarrow_R N \qquad M:A \mathbin{\&} B \Longrightarrow_E (\mathbf{fst}\,M, \mathbf{snd}\,M)$$

$$(\mathbf{case\ inl}\,M\ \mathbf{of\ inl}\,u \Rightarrow P \mid \mathbf{inr}\,v \Rightarrow Q) \Longrightarrow_R [M/u]P$$

$$(\mathbf{case\ inr}\,M\ \mathbf{of\ inl}\,u \Rightarrow P \mid \mathbf{inr}\,v \Rightarrow Q) \Longrightarrow_R [M/u]Q$$

$$M:A \oplus B \Longrightarrow_E (\mathbf{case}\,M\ \mathbf{of\ inl}\,u \Rightarrow \mathbf{inl}\,u \mid \mathbf{inl}\,v \Rightarrow \mathbf{inl}\,v)$$

$$M:\top \Longrightarrow_E () \qquad M:\mathbf{0} \Longrightarrow_E \mathbf{abort}\,M$$

Exponential rules The rules, local reductions, and local expansions for ! and ? are as follows.

$$\frac{\Gamma\,;\cdot \vdash M:A}{\Gamma\,;\cdot \vdash (!\,M):!A}\ !\,I \qquad \frac{\Gamma\,;\Delta \vdash M:!A \quad \Gamma, u:A\,;\Delta' \vdash P * J}{\Gamma\,;\Delta,\Delta' \vdash (\mathbf{let}\,!u = M\ \mathbf{in}\ P) * J}\ !\,E$$

$$\frac{\Gamma\,;\Delta \vdash E \div A\ poss}{\Gamma\,;\Delta \vdash ?E:?A}\ ?\,I \qquad \frac{\Gamma\,;\Delta \vdash M:?A \quad \Gamma\,;u:A \vdash E \div C\ poss}{\Gamma\,;\Delta \vdash (\mathbf{let}\,?u = M\ \mathbf{in}\ E) \div C\ poss}\ ?\,E$$

$$\mathbf{let}\,!u = !M\ \mathbf{in}\ P \Longrightarrow_R [M/u]P$$

$$M:!A \Longrightarrow_E \mathbf{let}\,!u = M\ \mathbf{in}\ !u$$

$$\mathbf{let}\,?u = ?E\ \mathbf{in}\ F \Longrightarrow_R \langle E/u \rangle F$$

$$M:?A \Longrightarrow_E \mathbf{let}\,?u = M\ \mathbf{in}\ ?u$$

Quantifier rules The rules, local reductions, and local expansions for ∀ and ∃ are as follows.

$$\frac{\Gamma\,;\Delta \vdash ([a/x]M):[a/x]A}{\Gamma\,;\Delta \vdash (\Lambda x.\ M):\forall x.A}\ \forall I^a \qquad \frac{\Gamma\,;\Delta \vdash M:\forall x.A}{\Gamma\,;\Delta \vdash (M \cdot t):[t/x]A}\ \forall E$$

$$\frac{\Gamma\,;\Delta \vdash ([t/x]M):[t/x]A}{\Gamma\,;\Delta \vdash [t,M]:\exists x.A}\ \exists I \qquad \frac{\Gamma\,;\Delta \vdash M:\exists x.A \quad \Gamma\,;\Delta', u:[a/x]A \vdash P * J}{\Gamma\,;\Delta,\Delta' \vdash (\mathbf{let}\,[a,u] = M\ \mathbf{in}\ P) * J}\ \exists E^a$$

$$(\Lambda x.\ M) \cdot t \Longrightarrow_R [t/x]M \qquad M:\forall x.A \Longrightarrow_E \Lambda x.\ M \cdot x$$

$$\mathbf{let}\,[a,u] = [t,M]\ \mathbf{in}\ P \Longrightarrow_R [t/a, M/u]P \qquad M:\exists x.A \Longrightarrow_E \mathbf{let}\,[a,u] = M\ \mathbf{in}\ [a,u]$$

Because we now have the full inventory of proof terms and expressions, we can give the full definition of the substitution operation. The definition of the expression substitution, $\langle E/u\rangle F$ is adapted from a similar definition for modal substitution in [94].

Definition 2.23 (Substitution)**.**

1. *The operation $[M/u]P$ is the standard capture-avoiding substitution of M for free occurrences of u in P. The notation $[M_1/u_1, \ldots, M_n/u_n]N$ is used to mean the M_i are substituted for the free u_i in N simultaneously, avoiding capture; this definition is also standard.*
2. *The operation $\langle E/u\rangle F$ is defined as follows:*

$$
\begin{aligned}
\langle M/u\rangle F &= [M/u]F \\
\langle \mathbf{let}\, v \otimes w = M\, \mathbf{in}\, E/u\rangle F &= \mathbf{let}\, v \otimes w = M\, \mathbf{in}\, \langle E/u\rangle F \\
\langle \mathbf{let} * = M\, \mathbf{in}\, E/u\rangle F &= \mathbf{let} * = M\, \mathbf{in}\, \langle E/u\rangle F \\
\langle \mathbf{case}\, M\, \mathbf{of}\, \mathbf{inl}\, v_1 \Rightarrow E_1 \mid \mathbf{inr}\, v_2 \Rightarrow E_2/u\rangle F &= \mathbf{case}\, M\, \mathbf{of}\, \mathbf{inl}\, v_1 \Rightarrow \langle E_1/u\rangle F \mid \mathbf{inr}\, v_2 \Rightarrow \langle E_2/u\rangle F \\
\langle \mathbf{abort}\, M/u\rangle F &= \mathbf{abort}\, M \\
\langle \mathbf{let}\, !\, v = M\, \mathbf{in}\, E/u\rangle F &= \mathbf{let}\, !\, v = M\, \mathbf{in}\, \langle E/u\rangle F \\
\langle \mathbf{let}\, ?\, v = M\, \mathbf{in}\, E/u\rangle F &= \mathbf{let}\, ?\, v = M\, \mathbf{in}\, \langle E/u\rangle F \\
\langle \mathbf{let}\, [a, v] = M\, \mathbf{in}\, E/u\rangle F &= \mathbf{let}\, [a, v] = M\, \mathbf{in}\, \langle E/u\rangle F
\end{aligned}
$$

This substitution is also understood to be capture-avoiding; capture can be avoided by first renaming the bound variables in E to be distinct from all free variables in F.

2.2.1 From the sequent calculus to natural deduction

In this section we shall prove the completeness of the natural deduction calculus with respect to the sequent calculus. As usual, this proof will be constructive, giving a process of transforming a sequent derivation into a natural deduction proof term.

Definition 2.24 (Sequent derivations to natural deduction proofs)**.** *We define the translation $\hookrightarrow$ between sequent derivations and natural deduction proofs as a derivation with the following rules.*

$$\frac{}{\mathrm{init}(u\!:\!p(\vec{t})) \hookrightarrow u}\ \hookrightarrow\!\mathrm{init}$$

$$\frac{\mathcal{D} \hookrightarrow P}{\mathrm{copy}(u.\, \mathcal{D}\,;v{:}A) \hookrightarrow [v/u]P} \hookrightarrow\mathrm{copy} \qquad \frac{\mathcal{D} \hookrightarrow P}{\mathrm{poss}(\mathcal{D}) \hookrightarrow P} \hookrightarrow\mathrm{poss}$$

$$\frac{\mathcal{D} \hookrightarrow M \quad \mathcal{D}' \hookrightarrow N}{\otimes R(\mathcal{D}, \mathcal{D}'\,; A \otimes B) \hookrightarrow M \otimes N} \hookrightarrow\otimes R \qquad \frac{\mathcal{D} \hookrightarrow P}{\otimes L(u.\, v.\, \mathcal{D}\,; w{:}A \otimes B) \hookrightarrow \mathbf{let}\, u \otimes v = w\, \mathbf{in}\, P} \hookrightarrow\otimes L$$

$$\frac{}{\mathbf{1}R \hookrightarrow *} \hookrightarrow\mathbf{1}R \qquad \frac{\mathcal{D} \hookrightarrow P}{\mathbf{1}L(\mathcal{D}\,; u{:}\mathbf{1}) \hookrightarrow \mathbf{let}\, * = u\, \mathbf{in}\, P} \hookrightarrow\mathbf{1}L$$

$$\frac{\mathcal{D} \hookrightarrow M}{-oR(u.\, \mathcal{D}\,; A \multimap B) \hookrightarrow \lambda u.\, M} \hookrightarrow\multimap R \qquad \frac{\mathcal{D} \hookrightarrow M \quad \mathcal{D}' \hookrightarrow N}{-oL(\mathcal{D}, u.\, \mathcal{D}'\,; v : A \multimap B) \hookrightarrow [v\, M/u]N} \hookrightarrow\multimap L$$

$$\frac{\mathcal{D} \hookrightarrow M \quad \mathcal{D}' \hookrightarrow N}{\& R(\mathcal{D}, \mathcal{D}'\,; A \,\&\, B) \hookrightarrow (M, N)} \hookrightarrow\& R$$

$$\frac{\mathcal{D} \hookrightarrow P}{\& L_1(u.\, \mathcal{D}\,; v{:}A \,\&\, B) \hookrightarrow [\mathbf{fst}\, v/u]P} \hookrightarrow\& L_1 \qquad \frac{\mathcal{D} \hookrightarrow P}{\& L_2(u.\, \mathcal{D}\,; v{:}A \,\&\, B) \hookrightarrow [\mathbf{snd}\, v/u]P} \hookrightarrow\& L_2$$

$$\frac{\mathcal{D} \hookrightarrow M}{\oplus R_1(\mathcal{D}\,; A \oplus B) \hookrightarrow \mathbf{inl}\, M} \hookrightarrow\oplus R_1 \qquad \frac{\mathcal{D} \hookrightarrow M}{\oplus R_2(\mathcal{D}\,; A \oplus B) \hookrightarrow \mathbf{inr}\, M} \hookrightarrow\oplus R_2$$

$$\frac{\mathcal{D} \hookrightarrow P \quad \mathcal{D}' \hookrightarrow Q}{\oplus L(u.\, \mathcal{D}, v.\, \mathcal{D}'\,; w{:}A \oplus B) \hookrightarrow \mathbf{case}\, w\, \mathbf{of\ inl}\, u \Rightarrow P \mid \mathbf{inr}\, v \Rightarrow Q} \hookrightarrow\oplus L$$

$$\frac{}{\top R \hookrightarrow ()} \hookrightarrow\top R \qquad \frac{}{\mathbf{0}L(u{:}\mathbf{0}) \hookrightarrow \mathbf{abort}\, u} \hookrightarrow\mathbf{0}L$$

$$\frac{\mathcal{D} \hookrightarrow M}{!R(\mathcal{D}\,; !A) \hookrightarrow !M} \hookrightarrow !R \qquad \frac{\mathcal{D} \hookrightarrow P}{!L(u.\, \mathcal{D}\,; v{:}!A) \hookrightarrow \mathbf{let}\, !u = v\, \mathbf{in}\, P} \hookrightarrow !L$$

$$\frac{\mathcal{D} \hookrightarrow E}{!R(\mathcal{D}\,; ?A) \hookrightarrow ?E} \hookrightarrow ?R \qquad \frac{\mathcal{D} \hookrightarrow E}{!L(u.\, \mathcal{D}\,; v{:}?A) \hookrightarrow \mathbf{let}\, ?u = v\, \mathbf{in}\, E} \hookrightarrow ?L$$

$$\frac{\mathcal{D} \hookrightarrow M}{\forall R(a.\, \mathcal{D}\,; \forall x.A) \hookrightarrow \Lambda x.\, [x/a]M} \hookrightarrow\forall R \qquad \frac{\mathcal{D} \hookrightarrow P}{\forall L(u.\, \mathcal{D}, t\,; v{:}\forall x.A) \hookrightarrow [v \cdot t/u]P} \hookrightarrow\forall L$$

$$\frac{\mathcal{D} \hookrightarrow M}{\exists R(\mathcal{D}, t\,; \exists x.A) \hookrightarrow [t, M]} \hookrightarrow\exists R \qquad \frac{\mathcal{D} \hookrightarrow P}{\exists L(a.\, u.\, \mathcal{D}\,; v{:}\exists x.A) \hookrightarrow \mathbf{let}\, [a, u] = v\, \mathbf{in}\, P} \hookrightarrow\exists L$$

Theorem 2.25 (Completeness of natural deduction). *If $\mathcal{D} :: \Gamma\,; \Delta \Longrightarrow J$, then there exists P such that $\mathcal{D} \hookrightarrow P$ and $\Gamma\,; \Delta \vdash P * J$.*

Proof. By induction on the structure of $\mathcal{D}$, using definition 2.24. For a representative case, consider

$$\frac{\mathcal{D}_1 :: \Gamma\,; \Delta, u_1{:}A \Longrightarrow J \quad \mathcal{D}_2 :: \Gamma\,; \Delta, u_2{:}B \Longrightarrow J}{\mathcal{D} = \oplus(u_1.\, \mathcal{D}_1, u_2.\, \mathcal{D}_2\,; v{:}A \oplus B) :: \Gamma\,; \Delta, v{:}A \oplus B \Longrightarrow J} \oplus L$$

$\left.\begin{array}{l}\mathcal{D}_1 \hookrightarrow P_1 \text{ and } \mathcal{D}_2 \hookrightarrow P_2 \text{ such that} \\ \Gamma\,; \Delta, u_1{:}A \vdash P_1 * J \text{ and } \Gamma\,; \Delta, u_2{:}B \vdash P_2 * J\end{array}\right\}$	i.h.
$\mathcal{D} \hookrightarrow \mathbf{case}\, v\, \mathbf{of\ inl}\, u_1 \Rightarrow P_1 \mid \mathbf{inr}\, u_2 \Rightarrow P_2$	definition 2.24

Then note that:

$$\dfrac{\dfrac{}{\Gamma\,;v{:}A\oplus B\vdash v{:}A\oplus B}\,\text{hyp}\quad \dfrac{}{\Gamma\,;\Delta,u_1:A\vdash P_1 * J}\,\text{i.h.}\quad \dfrac{}{\Gamma\,;\Delta,u_2:B\vdash P_2 * J}\,\text{i.h.}}{\Gamma\,;\Delta,v{:}A\oplus B\vdash(\textbf{case}\,v\,\textbf{of}\,\textbf{inl}\,u_1\Rightarrow P_1\mid\textbf{inr}\,u_2\Rightarrow P_2) * J}\,\oplus E \qquad \square$$

With theorem 2.25, we have fulfilled the principal reason for introducing the natural deduction calculus, which was to produce more perspicuous proof objects. The extraction of the natural deduction proof is in fact extremely systematic, requiring no appeals to complex procedures such as cut elimination on derivations. In the next section we shall go in the opposite direction, producing sequent derivations out of natural deduction proofs. This direction is considerably less systematic, requiring appeals to cut elimination at several points. This should not be surprising, because the sequent proofs in fact correspond to canonical–β-normal η-long–natural deduction proofs, but there are more natural deduction proofs than sequent proofs.

2.2.2 From natural deduction to the sequent calculus

In this section we will construct sequent derivations out of natural deduction proofs in a constructive fashion. It is not directly relevant for the purposes of building a theorem prover because we never search for natural deduction proof objects; however, it will establish the soundness of the natural deduction calculus with respect to the sequent calculus, validating our choice to present proofs in the natural deduction style.

Definition 2.26 (Natural deduction proofs to sequent derivations)**.** *We define the translation $\rightharpoonup$ from hypothetical natural deduction judgements to sequent derivations using the following inference rules.*

$$\dfrac{}{\Gamma\,;u{:}A\vdash u{:}A\rightharpoonup \text{id}(u{:}A)}\,{\rightharpoonup}\text{hyp}\qquad \dfrac{}{\Gamma,u{:}A\,;\cdot\vdash u{:}A\rightharpoonup \text{copy}(u.\ \text{id}(u:A)\,;w{:}A)}\,{\rightharpoonup}\text{hyp!}$$

$$\dfrac{\Gamma\,;\Delta\vdash M{:}A\ \mathit{res}\rightharpoonup\mathcal{D}}{\Gamma\,;\Delta\vdash M{:}A\ \mathit{poss}\rightharpoonup\text{poss}(\mathcal{D})}\,{\rightharpoonup}\text{poss}$$

$$\dfrac{\Gamma\,;\Delta\vdash M{:}A\rightharpoonup\mathcal{D}\quad \Gamma\,;\Delta'\vdash N{:}B\rightharpoonup\mathcal{D}'}{\Gamma\,;\Delta,\Delta'\vdash M\otimes N{:}A\otimes B\rightharpoonup \otimes R(\mathcal{D},\mathcal{D}'\,;A\otimes B)}\,{\rightharpoonup}\otimes I$$

$$\dfrac{\Gamma\,;\Delta\vdash M{:}A\otimes B\rightharpoonup\mathcal{D}\quad \Gamma\,;\Delta',u{:}A,v{:}B\vdash P * J\rightharpoonup\mathcal{E}}{\Gamma\,;\Delta,\Delta'\vdash(\textbf{let}\,u\otimes v=M\,\textbf{in}\,P) * J\rightharpoonup\mathcal{D}+_{w:A\otimes B}\otimes L(u.\,v.\,\mathcal{E}\,;w{:}A\otimes B)}\,{\rightharpoonup}\otimes E$$

$$\dfrac{\Gamma\,;\Delta,u{:}A\vdash M{:}B\rightharpoonup\mathcal{D}}{\Gamma\,;\Delta\vdash\lambda u.\,M{:}A\multimap B\rightharpoonup\multimap R(u.\,\mathcal{D}\,;A\multimap B)}\,{\rightharpoonup}{\multimap}I$$

$$\frac{\Gamma ; \Delta \vdash M{:}A \multimap B \rightharpoonup \mathcal{D} \quad \Gamma ; \Delta' \vdash N{:}A \rightharpoonup \mathcal{E}}{\Gamma ; \Delta, \Delta' \vdash (M\,N){:}B \rightharpoonup \mathcal{D} +_{w:A\multimap B} (\mathcal{E} +_{u:A} \multimap L(\mathrm{id}(u : A), v.\ \mathrm{id}(v : B) ; w : A \multimap B))} \rightharpoonup\multimap E$$

$$\frac{\Gamma ; \Delta \vdash M{:}A \rightharpoonup \mathcal{D} \quad \Gamma ; \Delta \vdash N{:}B \rightharpoonup \mathcal{D}'}{\Gamma ; \Delta \vdash (M, N){:}A \,\&\, B \rightharpoonup \&R(\mathcal{D}, \mathcal{D}' ; A \,\&\, B)} \rightharpoonup\&I$$

$$\frac{\Gamma ; \Delta \vdash M{:}A \,\&\, B \rightharpoonup \mathcal{D}}{\Gamma ; \Delta \vdash (\mathbf{fst}\,M){:}A \rightharpoonup \mathcal{D} +_{v:A\&B} \&L_1(u.\ \mathrm{id}(u : A) ; v : A \,\&\, B)} \rightharpoonup\&E_1$$

$$\frac{\Gamma ; \Delta \vdash M{:}A \,\&\, B \rightharpoonup \mathcal{D}}{\Gamma ; \Delta \vdash (\mathbf{snd}\,M){:}A \rightharpoonup \mathcal{D} +_{v:A\&B} \&L_2(u.\ \mathrm{id}(u : B) ; v : A \,\&\, B)} \rightharpoonup\&E_2$$

$$\frac{\Gamma ; \Delta \vdash M{:}A \rightharpoonup \mathcal{D}}{\Gamma ; \Delta \vdash (\mathbf{inl}\,M){:}A \oplus B \rightharpoonup \oplus R_1(\mathcal{D} ; A \oplus B)} \rightharpoonup\oplus I_1$$

$$\frac{\Gamma ; \Delta \vdash M{:}B \rightharpoonup \mathcal{D}}{\Gamma ; \Delta \vdash (\mathbf{inr}\,M){:}A \oplus B \rightharpoonup \oplus R_2(\mathcal{D} ; A \oplus B)} \rightharpoonup\oplus I_2$$

$$\frac{\Gamma ; \Delta \vdash M{:}A \oplus B \rightharpoonup \mathcal{D} \quad \Gamma ; \Delta', u{:}A \vdash P * J \rightharpoonup \mathcal{E}_1 \quad \Gamma ; \Delta', v{:}B \vdash Q * J \rightharpoonup \mathcal{E}_2}{\Gamma ; \Delta, \Delta' \vdash (\mathbf{case}\,M\,\mathbf{of}\,\mathbf{inl}\,u \Rightarrow P \mid \mathbf{inr}\,v \Rightarrow Q) * J \rightharpoonup \mathcal{D} +_{w:A\oplus B} \oplus L(u.\,\mathcal{E}_1, v.\,\mathcal{E}_2 ; w{:}A \oplus B)} \rightharpoonup\oplus E$$

$$\frac{}{\Gamma ; \Delta \vdash (){:}\top \rightharpoonup \top R} \rightharpoonup\top I \qquad \frac{\Gamma ; \Delta \vdash M{:}\mathbf{0} \rightharpoonup \mathcal{D}}{\Gamma ; \Delta \vdash (\mathbf{abort}\,M) * J \rightharpoonup \mathcal{D} +_{u:0} \mathbf{0}L(u{:}\mathbf{0})} \rightharpoonup\mathbf{0}E$$

$$\frac{\Gamma ; \cdot \vdash M{:}A \rightharpoonup \mathcal{D}}{\Gamma ; \cdot \vdash {!M}{:}{!A} \rightharpoonup {!R}(\mathcal{D} ; {!A})} \rightharpoonup{!I}$$

$$\frac{\Gamma ; \Delta \vdash M{:}{!A} \rightharpoonup \mathcal{D} \quad \Gamma, u{:}A ; \Delta' \vdash P * J \rightharpoonup \mathcal{E}}{\Gamma ; \Delta, \Delta' \vdash (\mathbf{let}\,{!u} = M\,\mathbf{in}\,P) * J \rightharpoonup \mathcal{D} +_{v:!A} {!L}(u.\,\mathcal{E} ; v{:}{!A})} \rightharpoonup{!E}$$

$$\frac{\Gamma ; \Delta \vdash E \div A\ poss \rightharpoonup \mathcal{D}}{\Gamma ; \Delta \vdash {?E}{:}{?A} \rightharpoonup {?R}(\mathcal{D} ; {?A})} \rightharpoonup{?I}$$

$$\frac{\Gamma ; \Delta \vdash M{:}{?A} \rightharpoonup \mathcal{D} \quad \Gamma ; u{:}A \vdash E \div C\ poss \rightharpoonup \mathcal{E}}{\Gamma ; \Delta \vdash (\mathbf{let}\,{?u} = M\,\mathbf{in}\,E) \div C\ poss \rightharpoonup \mathcal{D} +_{v:?A} {?L}(u.\,\mathcal{E} ; v{:}{?A})} \rightharpoonup{?E}$$

$$\frac{\Gamma ; \Delta \vdash [a/x]M{:}[a/x]A \rightharpoonup \mathcal{D}}{\Gamma ; \Delta \vdash (\Lambda x.\ M){:}\forall x.A \rightharpoonup \forall R(a.\ \mathcal{D} ; \forall x.A)} \rightharpoonup\forall I$$

$$\frac{\Gamma ; \Delta \vdash M{:}\forall x.A \rightharpoonup \mathcal{D}}{\Gamma ; \Delta \vdash (M \cdot t){:}[t/x]A \rightharpoonup \mathcal{D} +_{v:\forall x.A} \forall L(u.\ \mathrm{id}(u{:}[t/x]A), t ; v{:}\forall x.A)} \rightharpoonup\forall E$$

$$\frac{\Gamma ; \Delta \vdash M{:}[t/x]A \rightharpoonup \mathcal{D}}{\Gamma ; \Delta \vdash [t, M]{:}\exists x.A \rightharpoonup \exists R(\mathcal{D}, t ; \exists x.A)} \rightharpoonup\exists I$$

$$\frac{\Gamma ; \Delta \vdash M{:}\exists x.A \rightharpoonup \mathcal{D} \quad \Gamma ; \Delta', u{:}[a/x]A \vdash P * J \rightharpoonup \mathcal{E}}{\Gamma ; \Delta, \Delta' \vdash (\mathbf{let}\,[a, u] = M\,\mathbf{in}\,P) * J \rightharpoonup \mathcal{D} +_{v:\exists x.A} \exists L(a.\ u.\ \mathcal{E} ; v{:}\exists x.A)} \rightharpoonup\exists E$$

Theorem 2.27 (Soundness of natural deduction)**.**
If $\Gamma ; \Delta \vdash P * J$*, then there exists* $\mathcal{D}$ *such that* $\Gamma ; \Delta \vdash P * J \rightharpoonup \mathcal{D}$ *and* $\mathcal{D} :: \Gamma ; \Delta \Longrightarrow J$*.*

Proof. By induction on the structure of the given natural deduction proof. The following is a representative case.

$$\frac{\Gamma\,;\Delta \vdash M{:}A \multimap B \quad \Gamma\,;\Delta' \vdash N{:}A}{\Gamma\,;\Delta,\Delta' \vdash (M\,N){:}B}\multimap E$$

$\mathcal{D} :: \Gamma\,;\Delta \Longrightarrow A \multimap B$ and $\mathcal{E} :: \Gamma\,;\Delta' \vdash A$	i.h.
$\mathrm{id}(u : A) :: \Gamma\,; u{:}A \Longrightarrow A$	theorem 2.19
$\mathrm{id}(v : B) :: \Gamma\,; v{:}B \Longrightarrow B$	theorem 2.19
$\mathcal{F} = \multimap L(\mathrm{id}(u{:}A), v.\ \mathrm{id}(v{:}B)\,; w{:}A \multimap B) :: \Gamma\,; u{:}A, w{:}A \multimap B \Longrightarrow B$	$\multimap L$
$\mathcal{E} +_{u:A} \mathcal{F} :: \Gamma\,;\Delta', w{:}A \multimap B \Longrightarrow B$	theorem 2.21
$\mathcal{D} +_{w:A\multimap B} (\mathcal{E} +_{u:A} \mathcal{F}) :: \Gamma\,;\Delta,\Delta' \Longrightarrow B$	theorem 2.21

The remaining cases can be easily verified using definition 2.26. □

The computation in definition 2.26, although systematic, uses the powerful cut admissibility in nearly every case. Nearly every case therefore causes the overall sequent derivation to be globally rewritten. Nonetheless, we are mainly interested in the logical import of this translation, i.e., the following powerful completeness statement about the sequent calculus: every theorem in natural deduction has a cut free sequent calculus proof, and this statement itself can be shown constructively.

2.2.3 Normal forms

The natural deduction calculus of the previous section admits proofs that have unfinished local reductions inside them. The translation of sequent calculus derivations into natural deduction proofs, however, never introduces any redexes, and so the constructed proof term is β-normal. In this section, we formalise the notion of a normal natural deduction proof, and further give an algorithm for normalising proof terms. The algorithm presented for normalisation will be a straightforward constructive proof of the malleability of linear logic.

We follow the general schema outlined by Prawitz [97] by describing normal proofs as those that consist of two halves. One half reasons by deconstructing hypotheses into their component judgements using elimination rules; in this half, information flows from the premisses of an inference rule to the conclusion of the rule. The other half assembles information about the conclusion from information about the premisses using introduction rules; in this half the reasoning proceeds from the conclusion of a rule to its premisses. These two halves meet at a point where available information from the top half satisfies the reasoning required by the bottom half.

More formally, we annotate hypothetical judgements in natural deduction with a *directionality*:

$\Gamma\,;\Delta \vdash c \uparrow J$	normal derivations
$\Gamma\,;\Delta \vdash i \downarrow A\ \mathit{res}$	atomic derivations

We add a new judgemental rule of coercion that allows an atomic derivation to be a normal derivation. The rules of this calculus are summarized below; the syntax of normal and atomic proof terms, written using the meta variables $c, d, \ldots$ and $i, j, \ldots$ respectively, can be read off from the inference rules. As before, we adopt J to stand schematically for either the form $A\ \mathit{res}$ or $A\ \mathit{poss}$.

Judgemental rules

$$\frac{}{\Gamma\,; u{:}A \vdash u \downarrow A}\ \text{hyp} \qquad \frac{}{\Gamma, u : A\,; \cdot \vdash u \downarrow A}\ \text{hyp!}$$

$$\frac{\Gamma\,;\Delta \vdash c \uparrow A\ \mathit{res}}{\Gamma\,;\Delta \vdash c \uparrow A\ \mathit{poss}}\ \text{poss} \qquad \frac{\Gamma\,;\Delta \vdash i \downarrow A\ \mathit{res}}{\Gamma\,;\Delta \vdash i \uparrow A\ \mathit{res}}\ \text{coerce}$$

Multiplicative rules

$$\frac{\Gamma\,;\Delta_1 \vdash c \uparrow A \quad \Gamma\,;\Delta_2 \vdash d \uparrow B}{\Gamma\,;\Delta_1, \Delta_2 \vdash (c \otimes d) \uparrow A \otimes B}\ \otimes I \qquad \frac{\Gamma\,;\Delta_1 \vdash i \downarrow A \otimes B \quad \Gamma\,;\Delta_2, u{:}A, v{:}B \vdash c \uparrow J}{\Gamma\,;\Delta_1, \Delta_2 \vdash (\mathbf{let}\, u \otimes v = i\ \mathbf{in}\ c) \uparrow J}\ \otimes E$$

$$\frac{}{\Gamma\,; \cdot \vdash * \uparrow \mathbf{1}}\ \mathbf{1}I \qquad \frac{\Gamma\,;\Delta \vdash i \downarrow \mathbf{1} \quad \Gamma\,;\Delta' \vdash c \uparrow J}{\Gamma\,;\Delta, \Delta' \vdash (\mathbf{let}\, * = i\ \mathbf{in}\ c) \uparrow J}\ \mathbf{1}E$$

$$\frac{\Gamma\,;\Delta, u{:}A \vdash c \uparrow B}{\Gamma\,;\Delta \vdash (\lambda u.\, c) \uparrow A \multimap B}\ \multimap I \qquad \frac{\Gamma\,;\Delta \vdash i \downarrow A \multimap B \quad \Gamma\,;\Delta' \vdash c \uparrow A}{\Gamma\,;\Delta, \Delta' \vdash (i\ c) \downarrow B}\ \multimap E$$

Additive rules

$$\frac{\Gamma\,;\Delta \vdash c \uparrow A \quad \Gamma\,;\Delta \vdash d \uparrow B}{\Gamma\,;\Delta \vdash (c, d) \uparrow A \,\&\, B}\ \&I \qquad \frac{\Gamma\,;\Delta \vdash i \downarrow A \,\&\, B}{\Gamma\,;\Delta \vdash (\mathbf{fst}\, i) \downarrow A}\ \&E_1 \qquad \frac{\Gamma\,;\Delta \vdash i \downarrow A \,\&\, B}{\Gamma\,;\Delta \vdash (\mathbf{snd}\, i) \downarrow B}\ \&E_2$$

$$\frac{\Gamma ; \Delta \vdash c \uparrow A}{\Gamma ; \Delta \vdash (\mathbf{inl}\, c) \uparrow A \oplus B} \oplus I_1 \qquad \frac{\Gamma ; \Delta \vdash c \uparrow A}{\Gamma ; \Delta \vdash (\mathbf{inl}\, c) \uparrow A \oplus B} \oplus I_1$$

$$\frac{\Gamma ; \Delta \vdash i \downarrow A \oplus B \quad \Gamma ; \Delta', u{:}A \vdash c \uparrow J \quad \Gamma ; \Delta', v{:}B \vdash d \uparrow J}{\Gamma ; \Delta, \Delta' \vdash (\mathbf{case}\, i\, \mathbf{of}\, \mathbf{inl}\, u \Rightarrow c \mid \mathbf{inr}\, v \Rightarrow d) \uparrow J} \oplus E$$

$$\frac{}{\Gamma ; \Delta \vdash () \uparrow \top} \top I \qquad \frac{\Gamma ; \Delta \vdash i \downarrow \mathbf{0}}{\Gamma ; \Delta \vdash (\mathbf{abort}\, i) \downarrow J} \mathbf{0}E$$

Exponential rules

$$\frac{\Gamma ; \cdot \vdash c \uparrow A}{\Gamma ; \cdot \vdash (!\, c) \uparrow !A} \,!I \qquad \frac{\Gamma ; \Delta \vdash i \downarrow !A \quad \Gamma, u{:}A ; \Delta' \vdash c \uparrow J}{\Gamma ; \Delta, \Delta' \vdash (\mathbf{let}\, !u = i\, \mathbf{in}\, c) \uparrow J} \,!E$$

$$\frac{\Gamma ; \Delta \vdash c \uparrow A\ poss}{\Gamma ; \Delta \vdash ?c \uparrow ?A} \,?I \qquad \frac{\Gamma ; \Delta \vdash i \downarrow ?A \quad \Gamma ; u{:}A \vdash c \uparrow B\ poss}{\Gamma ; \Delta \vdash (\mathbf{let}\, ?u = i\, \mathbf{in}\, c) \uparrow B\ poss} \,?E$$

Quantifier rules

$$\frac{\Gamma ; \Delta \vdash ([a/x]c) \uparrow [a/x]A}{\Gamma ; \Delta \vdash (\Lambda x.\, c) \uparrow \forall x.A} \forall I^a \qquad \frac{\Gamma ; \Delta \vdash i \downarrow \forall x.A}{\Gamma ; \Delta \vdash (i \cdot t) \downarrow [t/x]A} \forall E$$

$$\frac{\Gamma ; \Delta \vdash ([t/x]c) \uparrow [t/x]A}{\Gamma ; \Delta \vdash [t, c] \uparrow \exists x.A} \exists I \qquad \frac{\Gamma ; \Delta \vdash i \downarrow \exists x.A \quad \Gamma ; \Delta', u{:}[a/x]A \vdash c \uparrow J}{\Gamma ; \Delta, \Delta' \vdash (\mathbf{let}\, [a, u] = i\, \mathbf{in}\, c) \uparrow J} \exists E^a$$

Theorem 2.28 (Substitution)**.**

1. *If* $\Gamma ; \Delta \vdash i \downarrow A$ *and*
   (a) $\Gamma ; \Delta', u{:}A \vdash c \uparrow J$*, then* $\Gamma ; \Delta, \Delta' \vdash [i/u]c \uparrow J$*.*
   (b) $\Gamma ; \Delta', u{:}A \vdash j \downarrow C$*, then* $\Gamma ; \Delta, \Delta' \vdash [i/u]j \downarrow C$*.*
2. *If* $\Gamma ; \cdot \vdash i \downarrow A$ *and*
   (a) $\Gamma, u{:}A ; \Delta \vdash c \uparrow J$*, then* $\Gamma ; \Delta \vdash [i/u]c \uparrow J$*.*
   (b) $\Gamma, u{:}A ; \Delta \vdash j \downarrow C$*, then* $\Gamma ; \Delta \vdash [i/u]j \downarrow C$*.*
3. *If* $\Gamma ; \Delta \vdash c \uparrow A\ poss$ *and* $\Gamma ; u{:}A \vdash d \uparrow C\ poss$*, then* $\Gamma ; \Delta \vdash \langle c/u \rangle\, d \uparrow C\ poss$*.*

Proof. Induction on the structure of the second derivation in (1) and (2), and on the first derivation in (3). □

Theorem 2.29 (Soundness of normal derivations)**.**

1. *If* $\Gamma ; \Delta \vdash c \uparrow J$*, then* $\Gamma ; \Delta \vdash c * J$*.*

2. *If* $\Gamma\,;\Delta \vdash i \downarrow C$*, then* $\Gamma\,;\Delta \vdash i{:}C$.

Proof. Induction on the structure of the given derivations. □

We already have enough machinery to prove the existence of normal forms of natural deduction proofs: we construct a sequent derivation using theorem 2.27, then use theorem 2.25 to give us a natural deduction proof term for that sequent derivation, then observe that definition 2.24 only creates normal proof terms.

Theorem 2.30 (Normal forms from sequent derivations)**.**
If $\mathcal{D} :: \Gamma\,;\Delta \Longrightarrow J$*, then* $\mathcal{D} \hookrightarrow c$ *such that* $\Gamma\,;\Delta \vdash c \uparrow J$.

Proof. Theorem 2.25 and inspection of definition 2.24. □

Theorem 2.31 (Existence of normal forms)**.**
If $\Gamma\,;\Delta \vdash P * J$*, then there exists a* c *such that* $\Gamma\,;\Delta \vdash c \uparrow J$.

Proof. Let $\mathcal{D}$ be such that $\Gamma\,;\Delta \vdash P * J \hookrightarrow \mathcal{D}$. By theorem 2.27, $\mathcal{D} :: \Gamma\,;\Delta \Longrightarrow J$. Then use theorem 2.30. □

It is possible to prove this more directly using a proof normalisation algorithm.

2.3 Historical review

A brief note on the genealogy of this presentation of linear logic. We trace the idea of dividing the hypotheses into an unrestricted and a linear zone back to Andreoli [7] for what he called a "dyadic system" for (classical) linear logic. This idea has seen considerable use since then: Hodas and Miller in the setting of logic programming in the uniform fragment [56], Benton *et al.* for linear term calculi [13]; more recently by Barber and Plotkin for the system DILL [11], Polakow and Pfenning for ordered logic [96, 95], and Howe in the setting of focused (backward) proof search for linear logic [57]. (The last of these unfortunately does not address the problem of focused proof-search in the two-zone setting, but rather uses it only to establish soundness and completeness of a one-zone focusing system, with explicit dereliction and promotion rules for modal contexts of the form $!\,\Gamma$.)

The natural deduction formulation of linear logic, particularly the form of the substitution (defn. 2.23), is essentially lifted from the judgemental reconstruction of modal logic by Pfenning and Davies [94].

The sequent calculus presented in this chapter with the ? connective is a first-order extension of JILL (Judgemental Intuitionistic Linear Logic) [27]. In JILL, it is possible to interpret the classical linear logic, classical affine logic (*i.e.*, CLL + arbitrary weakening), and the mysterious linear MIX rules (introduced by Girard [42]), using uniform parametric translations.

Chapter summary *In this chapter we have presented the (backward) sequent calculus for first-order intuitionistic linear logic, together with two modal extensions (truth and possibility), and proven the cut-elimination theorem. We have also presented a natural deduction formulation for this logic which is used to derive proofs from the sequent calculus for presentational purposes.*

The sequent calculus of this chapter will be the yardstick (for soundness and completeness) for the forward sequent calculi in subsequent chapters. The next chapter introduces the first of these forward calculi for the propositional fragment of the logic.

Chapter 3

Forward reasoning in the propositional fragment

We shall now begin our investigation into the use of the sequent calculus for automated reasoning in various fragments of linear logic. The first fragment we pick is the propositional fragment without possibility, i.e., $\{\otimes, \mathbf{1}, \&, \top, \oplus, \mathbf{0}, !\}$. The formulation in this chapter can be readily extended to include possibility.

We begin by examining the problem of *resource non-determinism* in the backward sequent calculus presented in chapter 2.1, where we start with a given goal sequent and use the inference rules of the logic in the backward direction in order to refine the goals until we are left with axiomatic or initial sequents. Because this search direction starts from the goal sequent, it is sometimes also called "goal-directed" search.

The purpose of the backward search strategy will be to present a key difficulty, *multiplicative non-determinism*, and motivate a forward search strategy that avoids this difficulty. The forward search strategy will use a forward version of the sequent calculus. It shall start from known facts, and iteratively use the rules of the forward calculus to generate new facts, with the goal of eventually discovering a proof of the goal. The kind of forward reasoning used—the inverse method [73, 116]—also merits the description "goal-directed" because it restricts all rule applications to subformulas of the goal sequent, using a strong subformula property of the sequent calculus.

3.1 Resource management

The novelty in automated reasoning in linear logic lies in handling resources efficiently – the *resource management problem*. We can trace the origin of this problem to the lack of structural weakening and contraction; indeed, without these rules linear logic with multiplicatives, additives and exponentials becomes undecidable, even for the propositional case. We can recognize the following two classes of resource management problems.

Structural non-determinism, which occurs for unrestricted resources, and the linear resources in the rules for the additive units, such as $\top$:

$$\frac{}{\Gamma\,;\Delta \Longrightarrow \top}$$

For these rules, the conclusion sequent contains resources that do not occur structurally in the (possibly non-existent) premisses. Thus, a forward reading of these rules has to invent this context using extra-logical means, a futile approach in general because of the lack of a decision procedure. Fortunately, a clean solution exists for this problem, which we explain in section 3.2.2. Note that structural non-determinism is completely absent in the backward direction because of the *subformula property*: all elements in the premiss of an inference rule occur in the conclusion of the rule, possibly as subformulas.

Multiplicative non-determinism, which arises from multiplicative rules with more than one premiss, for example for $\otimes R$:

$$\frac{\Delta \Longrightarrow A \quad \Delta' \Longrightarrow B}{\Delta, \Delta' \Longrightarrow A \otimes B}$$

Absent weakening, in the backward direction such rules must infer a division (into Δ and Δ' above) of the linear resources of the conclusion to distribute into the premisses. Note that this kind of non-determinism does not exist in a forward reading, where we simply conjoin the resources of the premisses to construct the conclusion.

In the domain of top-down linear logic programming—refining goals by applying inference rules in the *backward* direction until they become initial (eg. *Lolli* [55] or *Lygon* [118])—approaches to combating this kind of non-determinism fall into two broad kinds. The first kind commit to an input-output interpretation of hypotheses. For the $\otimes R$ rule for example, proof search proceeds eagerly along the first premiss until it reaches the initial sequents with some unconsumed resources. These unconsumed resources then form the linear context for the second branch of the derivation tree corresponding to the second premiss. Proof search therefore becomes completely deterministic, though not free of complications. For example, when attempting to prove $\top \otimes A$, the first branch can consume an arbitrary number of resources; thus an unprincipled implementation of the input-output idea will continue to involve a potentially exponential number of backtracking operations. As a possible answer to such complications, Cervesato *et al.* [23] refine the sequent judgement with Boolean "strictness" flags and add a context of *strict* resources, which adequately solves the resource management problems for linear logic programming.

Approaches of the second kind perform general search with constraint solving. For example, in [49], Boolean flags mark uses of resources, with inference rules guarded by constraints on these Boolean flags. Particular proof strategies then correspond to particular solutions for these constraint problems. In fact, we may view the first kind of approach as a kind of solution to the constraint problem, where the Boolean constraints encode the input-output interpretation. Without detailing such constraint-based resource management systems, we refer to the work of Harland, Pym and Winikoff, now almost a decade old [48, 50].

Interestingly, the rules for the additive units present significant problems in the backward direction also. For the input output interpretation, the additive unit $\top$ can "consume" an arbitrary number of resources in its branch of the proof. Thus, proving $\top \otimes A$ may require backtracking if search proceeds down the $\top$ branch first without determining the number of resources needed for the other branch. For a more complete discussion, see [23]. The problem of structural non-determinism thus exists in both forward and backward reasoning, but the nature—*invention* of unknown resources in the forward direction, and *allocation/garbage-collection* of resources in the backward direction—differs sufficiently that we cannot immediately adapt resource management approaches for the latter to the former.

Other non-deterministic choices do exist during proof search, but they do not share the peculiar nature of resource management problems, and certainly occur for ordinary (non-linear) logic also. For example, *disjunctive non-determinism* for connectives with multiple introduction rules (on the left or right); *conjunctive non-determinism* for multi-premiss rules, where the order of exploration affects search in significant ways. In the forward direction, conjunctive non-determinism arises from saturation-based—i.e., fair—search, a necessity to ensure completeness, and various other possibilities. Because of the standard nature of these problems, we refer readers to the Handbook article on the inverse method [35].

3.2 Forward Sequent Calculus

3.2.1 Multiplicative exponential linear logic

First let us consider the problem of multiplicative non-determinism. As noted earlier, multiplicative resource non-determinism does not occur at all in the forward reading of such inference rules, where we start with the sequents involving linear contexts Δ and Δ', and conclude a sequent involving Δ, Δ'. For the purely multiplicative fragment of linear logic, i.e., $\otimes$, $\mathbf{1}$ and $\multimap$, the backward rules are, in fact, already sufficient for forward reasoning, so we add the first complicating factor: the ! exponential. This fragment is sometimes called MELL (multiplicative exponential linear logic), and as of this writing the decision problem for it is unknown, though it is at least NP-hard.[1]

This fragment, although severely restricted because of the lack of alternation or any way of expressing choice, is expressive enough for a variety of uses. Let us consider a motivating example of coin changing, where we describe the operation of turning two nickels into a dime as:

$$\mathtt{nickel} \otimes \mathtt{nickel} \multimap \mathtt{dime} \tag{R_1}$$

Some more simple rules for coin transformations:

$$\mathtt{nickel} \multimap \mathtt{penny} \otimes \mathtt{penny} \otimes \mathtt{penny} \otimes \mathtt{penny} \otimes \mathtt{penny} \tag{R_1}$$

$$\mathtt{quarter} \multimap \mathtt{dime} \otimes \mathtt{dime} \otimes \mathtt{nickel} \tag{R_2}$$

$$\mathtt{dollar} \multimap \mathtt{quarter} \otimes \mathtt{quarter} \otimes \mathtt{quarter} \otimes \mathtt{quarter} \tag{R_3}$$

We may then ask a query of the form:

$$\mathtt{quarter} \otimes \mathtt{nickel} \multimap \mathtt{dime} \otimes \mathtt{dime} \otimes \mathtt{dime} \tag{C}$$

[1]This is based on the complexity of the $\{\otimes, \mathbf{1}, \multimap\}$ fragment, which can be used to encode Petri-net reachability, which in turn can be verified in polynomial time.

which we intend to prove using the transition rules R_1, R_2, R_3 and R_4 any number of times. In other words, our goal sequent is:

$$\underbrace{R_1, R_2, R_3, R_4}_{\Gamma_0} ; \cdot \Longrightarrow C$$

The derivation of the above goal sequent is:

$$\dfrac{\dfrac{}{\Gamma_0 ; \mathtt{q} \Longrightarrow \mathtt{q}}\;\text{init} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Gamma_0 ; \mathtt{n}, \mathtt{n} \Longrightarrow \mathtt{n} \otimes \mathtt{n}}\;\otimes R \qquad \dfrac{\vdots}{\Gamma_0 ; \mathtt{d}, \mathtt{d}, \mathtt{d} \Longrightarrow \mathtt{d} \otimes \mathtt{d} \otimes \mathtt{d}}\;\otimes R}{\Gamma_0 ; \mathtt{n}, \mathtt{d}, \mathtt{d}, \mathtt{n}, \mathtt{n} \otimes \mathtt{n} \multimap \mathtt{d} \Longrightarrow \mathtt{d} \otimes \mathtt{d} \otimes \mathtt{d}}\;\multimap R}{\Gamma_0 ; \mathtt{n}, \mathtt{d}, \mathtt{d}, \mathtt{n} \Longrightarrow \mathtt{d} \otimes \mathtt{d} \otimes \mathtt{d}}\;\text{copy: } R_1}{\Gamma_0 ; \mathtt{n}, \mathtt{d} \otimes \mathtt{d} \otimes \mathtt{n} \Longrightarrow \mathtt{d} \otimes \mathtt{d} \otimes \mathtt{d}}\;\otimes L, \otimes L}{\dfrac{\dfrac{\dfrac{\Gamma_0 ; \mathtt{q}, \mathtt{n}, \mathtt{q} \multimap \mathtt{d} \otimes \mathtt{d} \otimes \mathtt{n} \Longrightarrow \mathtt{d} \otimes \mathtt{d} \otimes \mathtt{d}}{\Gamma_0 ; \mathtt{q}, \mathtt{n} \Longrightarrow \mathtt{d} \otimes \mathtt{d} \otimes \mathtt{d}}\;\text{copy: } R_3}{\Gamma_0 ; \mathtt{q} \otimes \mathtt{n} \Longrightarrow \mathtt{d} \otimes \mathtt{d} \otimes \mathtt{d}}\;\otimes L}{\Gamma_0 ; \cdot \Longrightarrow C}\;\multimap R}\;\multimap L$$

We use the single sequent arrow $\longrightarrow$ to represent sequents in the forward direction, but the structure and judgements carry over from the backward calculus of chapter 2.1. In the forward direction, the initial sequents cannot construct the unrestricted contexts because the "init" rule has no premisses. Forward initial sequents therefore leave the unrestricted zone blank:

$$\dfrac{}{\cdot ; p \longrightarrow p}\;\text{init}$$

In the "copy" rule in the forward direction, we can no longer assume that the copied resource is already present in the unrestricted context. We therefore, simply add it to the unrestricted context.

$$\dfrac{\Gamma ; \Delta, A \longrightarrow C}{\Gamma, A ; \Delta \longrightarrow C}\;\text{copy}$$

Of course, if the same resource was copied twice, then we will end up with two versions of A in the unrestricted zone. Thus we add an explicit rule of factoring (contraction) in the forward direction:

$$\dfrac{\Gamma, A, A ; \Delta \longrightarrow C}{\Gamma, A ; \Delta \longrightarrow C}\;\text{factor}$$

We record factoring in the derivations using the syntax factor($u.\ v.\ \mathcal{D}_f\ ;w : A$), where u and v are the bound labels for the unrestricted resources in $\mathcal{D}$ that are factored into $w : A$.

Multiple branches of the derivation will therefore have different compositions of the unrestricted context, so in a binary rule it is no longer possible to require the unrestricted contexts in the two premisses to be identical. We remedy this by combining the unrestricted contexts of the premisses multiplicatively, and let factoring take care of duplicates.

$$\frac{\Gamma\,;\Delta \longrightarrow A \quad \Gamma'\,;\Delta' \longrightarrow B}{\Gamma,\Gamma'\,;\Delta,\Delta' \longrightarrow A \otimes B}\ \otimes R$$

The example derivation from before in the forward direction is therefore as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{R_1\,;\mathtt{q} \longrightarrow \mathtt{q}}\ \text{init} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\cdot\,;\mathtt{n},\mathtt{n} \longrightarrow \mathtt{n}\otimes\mathtt{n}}\ \otimes R \quad \dfrac{}{\cdot\,;\mathtt{d},\mathtt{d},\mathtt{d} \longrightarrow \mathtt{d}\otimes\mathtt{d}\otimes\mathtt{d}}\ \otimes R}{\cdot\,;\mathtt{n},\mathtt{d},\mathtt{d},\mathtt{n},\mathtt{n}\otimes\mathtt{n} \multimap \mathtt{d} \longrightarrow \mathtt{d}\otimes\mathtt{d}\otimes\mathtt{d}}\ {\multimap}R}{R_1\,;\mathtt{n},\mathtt{d},\mathtt{d},\mathtt{n} \longrightarrow \mathtt{d}\otimes\mathtt{d}\otimes\mathtt{d}}\ \text{copy: } R_1}{R_1\,;\mathtt{n},\mathtt{d}\otimes\mathtt{d}\otimes\mathtt{n} \longrightarrow \mathtt{d}\otimes\mathtt{d}\otimes\mathtt{d}}\ \otimes L}{R_1\,;\mathtt{q},\mathtt{n},\mathtt{q} \multimap \mathtt{d}\otimes\mathtt{d}\otimes\mathtt{n} \longrightarrow \mathtt{d}\otimes\mathtt{d}\otimes\mathtt{d}}\ {\multimap}L}{R_1,R_3\,;\mathtt{q},\mathtt{n} \longrightarrow \mathtt{d}\otimes\mathtt{d}\otimes\mathtt{d}}\ \text{copy: } R_3}{\dfrac{R_1,R_3\,;\mathtt{q}\otimes\mathtt{n} \longrightarrow \mathtt{d}\otimes\mathtt{d}\otimes\mathtt{d}}{R_1,R_3\,;\cdot \longrightarrow C}\ {\multimap}R}\ \otimes L$$

Now note that the sequent $R_1, R_3\,; \cdot \Longrightarrow C$ is stronger (in the sense of theorem 2.17) than our required goal sequent $R_1, R_2, R_3, R_4\,; \cdot \Longrightarrow C$.

The $\otimes L$ rule in the forward direction requires a brief note. If the unrestricted resource A is present in the premiss, then we remove it from the unrestricted context in the conclusion, as expected.

$$\frac{\Gamma, A\,;\Delta \Longrightarrow C}{\Gamma\,;\Delta, !A \Longrightarrow C}$$

However, it is possible that the resource A is not present in the premiss; in this case case we recall that a forward sequent stands for all its weakened forms, and therefore implicitly "weaken" the premiss to produce the required conclusion:

$$\frac{\Gamma\,;\Delta \Longrightarrow C \quad A \notin \Gamma}{\Gamma\,;\Delta, !A \Longrightarrow C}$$

These two forms of the $!L$ rule can be written together as follows, where the notation $\Gamma \backslash A$ is the usual multiset difference, i.e., it denotes the operation of removing A from Γ if it exists there.

$$\frac{\Gamma\,;\Delta \Longrightarrow C \quad A \notin \Gamma}{\Gamma \backslash A\,;\Delta, !A \Longrightarrow C}$$

Summary of the formal system

We pick the following propositional fragment of linear logic.

$$A, B, \ldots \quad ::= \quad p \mid A \otimes B \mid \mathbf{1} \mid A \multimap B \mid !A$$

Forward sequents have the following shape:

$$\underbrace{u_1 : (A_1\ \mathit{unr}), \ldots, u_m : (A_m\ \mathit{unr})}_{\Gamma}\ ;\ \underbrace{v_1 : (B_1\ \mathit{res}), \ldots, v_n : (B_n\ \mathit{res})}_{\Delta} \longrightarrow C\ \mathit{goal}.$$

As usual, we leave the hypotheses labels u, v, etc. and the judgemental labels *unr*, *res* and *goal* out for brevity. We adopt the same syntax of derivations as in the backward calculus of section 2.1.5, but distinguish forward and backward derivations with a subscript f or b, respectively. The only rule in the forward direction that does not have a corresponding backward rule is "factor", for which we have an additional syntactic form for the forward direction. Our sequent calculus has the following rules.

Judgemental rules

$$\frac{}{\cdot\,; p \longrightarrow p}\ \text{init} \qquad \frac{\Gamma\,;\Delta, A \longrightarrow C}{\Gamma, A\,;\Delta \longrightarrow C}\ \text{copy} \qquad \frac{\Gamma, A, A\,;\Delta \longrightarrow C}{\Gamma, A\,;\Delta \longrightarrow C}\ \text{factor}$$

Logical rules

$$\frac{\Gamma\,;\Delta \longrightarrow A \quad \Gamma'\,;\Delta' \longrightarrow B}{\Gamma, \Gamma'\,;\Delta, \Delta' \longrightarrow A \otimes B}\ \otimes R \qquad \frac{\Gamma\,;\Delta, A, B \longrightarrow C}{\Gamma\,;\Delta, A \otimes B \longrightarrow C}\ \otimes L$$

$$\frac{}{\cdot\,;\cdot \longrightarrow \mathbf{1}}\ \mathbf{1}R \qquad \frac{\Gamma\,;\Delta \longrightarrow C}{\Gamma\,;\Delta, \mathbf{1} \longrightarrow C}\ \mathbf{1}L$$

$$\frac{\Gamma\,;\Delta, A \longrightarrow B}{\Gamma\,;\Delta \longrightarrow A \multimap B}\ {\multimap}R \qquad \frac{\Gamma\,;\Delta \longrightarrow A \quad \Gamma'\,;\Delta', B \longrightarrow C}{\Gamma, \Gamma'\,;\Delta, \Delta', A \multimap B \longrightarrow C}\ {\multimap}L$$

$$\frac{\Gamma\,;\cdot \longrightarrow A}{\Gamma\,;\cdot \longrightarrow\, !A}\;!R \qquad \frac{\Gamma\,;\Delta \longrightarrow C}{\Gamma\backslash A\,;\Delta, !A \longrightarrow C}\;!L$$

Definition 3.1 (Forward derivations to backward derivations)**.** *The translation* $(-)^o$ *from forward to backward derivations mimics the structure of derivations exactly, except for the "factor" rule, for which we have:*

$$(\text{factor}(u.\ v.\ \mathcal{D}_f\,;w{:}A))^o = [w/u, w/v](\mathcal{D}_f)^o$$

Theorem 3.2 (Soundness of forward derivations)**.**
If $\mathcal{D}_f :: \Gamma\,;\Delta \longrightarrow C$*, then* $(\mathcal{D}_f)^o :: \Gamma\,;\Delta \Longrightarrow C$*.*

Proof. Induction on the structure of $\mathcal{D}_f$. The following are two representative cases.

Case.

$$\frac{\mathcal{D}_f :: \Gamma\,;\Delta \longrightarrow A \quad \mathcal{D}'_f :: \Gamma'\,;\Delta', u{:}B \longrightarrow C}{\multimap L(\mathcal{D}_f, u.\ \mathcal{D}'_f\,; v{:}A \multimap B) :: \Gamma,\Gamma'\,;\Delta,\Delta', v : A \multimap B \longrightarrow C}\;\multimap L$$

$(\mathcal{D}_f)^o :: \Gamma\,;\Delta \Longrightarrow A$ — i.h.
$(\mathcal{D}_f)^o :: \Gamma,\Gamma'\,;\Delta \Longrightarrow A$ — theorem 2.17
$(\mathcal{D}'_f)^o :: \Gamma,\Gamma'\,;\Delta', u{:}B \Longrightarrow C$ — similarly
$\multimap L((\mathcal{D}_f)^o, u.\ (\mathcal{D}'_f)^o\,; v{:}A \multimap B) :: \Gamma,\Gamma'\,;\Delta,\Delta', v{:}A \multimap B \Longrightarrow C$ — $\multimap L$

Case.

$$\frac{\mathcal{D}_f :: \Gamma, u : A, v : A\,;\Delta \longrightarrow C}{\text{factor}(u.\ v.\ \mathcal{D}_f\,; w : A) :: \Gamma, w : A\,;\Delta \longrightarrow C}\;\text{factor}$$

$(\mathcal{D}_f)^o :: \Gamma, u : A, v : A\,;\Delta \Longrightarrow C$ — i.h.
$\mathcal{E} = \text{copy}(z.\ \text{id}(z : A)\,; w{:}A) :: \Gamma, w : A\,;\cdot \Longrightarrow A$
$\mathcal{E} +_{v:A} (\mathcal{E} +_{u:A} (\mathcal{D}_f)^o) :: \Gamma, w : A\,;\Delta \Longrightarrow C$ □

The completeness theorem cannot be shown in such a clean and constructive manner because there are several forward derivations for a given backward derivation and no canonical way to translate a backward to a forward derivation. Instead, we prove it as an existential property.

Theorem 3.3 (Completeness of forward derivations)**.**
If $\Gamma\,;\Delta \Longrightarrow C$*, then* $\Gamma'\,;\Delta \longrightarrow C$ *for some* $\Gamma' \subseteq \Gamma$*.*

Proof. Induction on the structure of the derivation of $\mathcal{D}_b :: \Gamma\,;\Delta \Longrightarrow C$. The following is a representative case.

$$\frac{\mathcal{D}_{b1} :: \Gamma\,;\Delta \Longrightarrow A \quad \mathcal{D}_{b2} :: \Gamma\,;\Delta', u{:}A \Longrightarrow C}{{\multimap}L(\mathcal{D}_{b1}, u.\,\mathcal{D}_{b2}\,; v{:}A \multimap B) :: \Gamma\,;\Delta,\Delta', v{:}A \multimap B \Longrightarrow C}\ {\multimap}L$$

We know that $\Gamma_1\,;\Delta \longrightarrow A$ and $\Gamma_2\,;\Delta', u{:}B \longrightarrow C$ for some $\Gamma_1 \subseteq \Gamma$ and $\Gamma_2 \subseteq \Gamma$ by the induction hypotheses. By ${\multimap}L$, therefore, $\Gamma_1, \Gamma_2\,;\Delta,\Delta', v{:}A \multimap B \longrightarrow C$. Because both $\Gamma_1 \subseteq \Gamma$, and $\Gamma_2 \subseteq \Gamma$, we use "factor" to merge the duplicate propositions in in Γ_1 and Γ_2 to obtain a context $\Gamma' \subseteq \Gamma$ for which $\Gamma'\,;\Delta \Longrightarrow C$. □

3.2.2 Extending with the additive connectives

As mentioned before, the additive units $\top$ and $\mathbf{0}$ cause a similar problem with the linear context also:

$$\frac{}{\Gamma\,;\Delta \Longrightarrow \top}\ \top R \qquad \frac{}{\Gamma\,;\Delta, \mathbf{0} \Longrightarrow C}\ \mathbf{0}L$$

The arbitrary linear contexts Δ (and side formula C) do not occur in the (non-existent) premisses, and can therefore not be deterministically constructed. However, linearity prevents us from writing simply

$$\frac{}{\cdot\,;\cdot \longrightarrow \top}$$

because the linear context can not be weakened as needed. (For instance, $\cdot\,;\top \longrightarrow \top$ is not derivable with this rule.)

We solve this problem by constructing the unknown portions of sequents as needed, adapting the scheme laid out in the previous section for the unrestricted resources, allowing weakening of the linear resources for those sequents for which it is admissible.

Definition 3.4 (Weak backward sequents)**.** *A sequent* $\Gamma\,;\Delta \Longrightarrow C$ *is said to be weak if, assuming it is valid, the sequent* $\Gamma\,;\Delta' \Longrightarrow C$ *is also valid for all* $\Delta' \supseteq \Delta$*. A sequent that is not weak is a strong sequent.*

In other words, the following rule is admissible for weak sequents.

$$\frac{\Gamma\,;\Delta \Longrightarrow C}{\Gamma\,;\Delta,\Delta' \Longrightarrow C}\ \text{linear-weaken}$$

The additive atoms give rise to weak sequents: $\Gamma\,;\Delta \Longrightarrow \top$ and $\Gamma\,;\Delta, 0 \Longrightarrow C$. More generally, having a conjunctive $\top$ on the right or a conjunctive **0** on the left will make the sequent weak.

In the forward direction we explicitly keep track of which sequents are weak by means of a Boolean flag on the linear context.

Definition 3.5 (Forward sequents)**.** *A forward sequent has one of the following two forms:*

$$\begin{array}{ll} \Gamma\,;[\Delta]_0 \longrightarrow C & \textit{strong sequent} \\ \Gamma\,;[\Delta]_1 \longrightarrow \gamma & \textit{weak sequent} \end{array}$$

Here, γ has one of the forms $\cdot$ or C. We will slightly abuse notation by writing $\Gamma\,;[\Delta]_w \longrightarrow \gamma$ to stand for either of the above two forms, using w as a meta-variable for the weakness flag, keeping in mind that $\Gamma\,;[\Delta]_0 \longrightarrow \cdot$ is a disallowed form.

The correspondence between forward and backward sequents is stated in terms of soundness.

Definition 3.6 (Soundness of forward sequents)**.**

1. *A strong sequent $\Gamma\,;[\Delta]_0 \longrightarrow C$ is said to be sound if $\Gamma'\,;\Delta \Longrightarrow C$ for any $\Gamma' \supseteq \Gamma$.*
2. *A weak sequent $\Gamma\,;[\Delta]_1 \longrightarrow \gamma$ is said to be sound if $\Gamma'\,;\Delta' \Longrightarrow C$ for any $\Gamma' \supseteq \Gamma$, $\Delta' \supseteq \Delta$ and $C \supseteq \gamma$.*

We use $C \supseteq \gamma$ to indicate that either $\gamma = \cdot$ or $\gamma = C$.

The final ingredient required is a *subsumption* or "weaker than" relation between forward sequents.

Definition 3.7 (Subsumption)**.**
We define the $\prec$ relation between forward sequents as follows.

$$\begin{array}{ll} (\Gamma\,;[\Delta]_0 \longrightarrow C) \prec (\Gamma'\,;[\Delta]_0 \longrightarrow C) & \textit{if } \Gamma' \supseteq \Gamma \\ (\Gamma\,;[\Delta]_1 \longrightarrow \gamma) \prec (\Gamma'\,;[\Delta']_w \longrightarrow \gamma') & \textit{if } \Gamma' \supseteq \Gamma,\ \Delta' \supseteq \Delta, \textit{ and } \gamma' \supseteq \gamma \end{array}$$

For binary rules, the combination of linear zones will depend on whether the corresponding sequent is weak or strong. Consider the backward &*R* rule which is additive:

$$\frac{\Gamma\,;\Delta \Longrightarrow A \quad \Gamma\,;\Delta' \Longrightarrow B}{\Gamma\,;\Delta \Longrightarrow A \,\&\, B}\ \&R$$

If both premisses are strong in the forward direction, then this rule can be directly adapted to the forward direction.

$$\frac{\Gamma \,;\, [\Delta]_0 \longrightarrow A \quad \Gamma' \,;\, [\Delta]_0 \longrightarrow B}{\Gamma, \Gamma' \,;\, [\Delta]_0 \longrightarrow A \mathbin{\&} B}\ \&R$$

If one premiss is weak and the other strong, the weak resources must be a subset of the strong resources to remain consistent with definition 3.7.

$$\frac{\Gamma \,;\, [\Delta]_0 \longrightarrow A \quad \Gamma' \,;\, [\Delta']_1 \longrightarrow B \quad (\Delta' \subseteq \Delta)}{\Gamma, \Gamma' \,;\, [\Delta]_0 \longrightarrow A \mathbin{\&} B}$$

If both premisses are weak, then the conclusion is also weak, but what resources are present in the conclusion? In the ground case, we can simply take the maximal multiplicity for each proposition on the two premisses, which we write using $\sqcup$; in other words, if a resource A occurs m times in Δ and n times in Δ', then it occurs $max(m + n)$ times in $\Delta \sqcup \Delta'$. To see that this is sound, simply apply weakening to add the missing copies, equalizing the linear contexts in the premisses. It is also complete because the maximum represents the least upper bound.

$$\frac{\Gamma \,;\, [\Delta]_1 \longrightarrow A \quad \Gamma' \,;\, [\Delta']_1 \longrightarrow B}{\Gamma, \Gamma' \,;\, [\Delta \sqcup \Delta']_1 \longrightarrow A \mathbin{\&} B}$$

Fortunately, we do not need to generalise the $\&R$ rules any further to allow weakening on the right also. Consider, for instance, the following candidate:

$$\frac{\Gamma \,;\, [\Delta]_0 \longrightarrow A \quad \Gamma \,;\, [\Delta']_1 \longrightarrow \cdot \quad (\Delta' \subseteq \Delta)}{\Gamma, \Gamma' \,;\, [\Delta]_1 \longrightarrow A \mathbin{\&} B}$$

Here, the conclusion sequent is simply a weakened form of the second premiss, and therefore is entirely redundant as it will be immediately subsumed in the inverse method loop (see sec. 4.1.6 for details of this loop).

In the above forms of the $\&R$ rule, the difference is in the ways in which the linear contexts of the input premisses are allowed to be combined. To ease the presentation of the rules and also to foreshadow the kind of constructions we will require in the first-order case (in chapter 5), we make a few definitions.

Definition 3.8 (Additive composition). *Given two linear contexts with weakness flags, $[\Delta]_w$ and $[\Delta]_{w'}$, we define the additive composition of the contexts, written $[\Delta]_w + [\Delta']_{w'}$ as follows:*

$$[\Delta]_w + [\Delta']_{w'} = \begin{cases} [\Delta]_0 & \text{if } w = w' = 0 \text{ and } \Delta = \Delta' \\ [\Delta]_0 & \text{if } w = 0, w' = 1 \text{ and } \Delta' \subseteq \Delta \\ [\Delta']_0 & \text{if } w = 1, w' = 0 \text{ and } \Delta \subseteq \Delta' \\ [\Delta \sqcup \Delta']_1 & \text{if } w = w' = 1 \end{cases}$$

Note that the additive composition is a partial function.

We can then write the $\&R$ in a concise form:

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow A \quad \Gamma'\,;[\Delta']_{w'} \longrightarrow B}{\Gamma, \Gamma'\,;[\Delta]_w + [\Delta']_{w'} \longrightarrow A \,\&\, B}\ \&R$$

The implicit understanding is that if the additive composition in the conclusion is not defined, then the rule is not applicable.

On the left, we have to include some additional cases for when the required resources are not actually present in a weak premiss. As an illustrative example, consider $\otimes L$:

$$\frac{\Gamma\,;\Delta, A, B \Longrightarrow C}{\Gamma\,;\Delta, A \otimes B \Longrightarrow C}\ \otimes L$$

In the forward direction, if both A and B are present in the premiss, then the weakness of the premiss carries through to the conclusion.

$$\frac{\Gamma\,;[\Delta, A, B]_0 \longrightarrow \gamma}{\Gamma\,;[\Delta, A \otimes B]_0 \longrightarrow \gamma} \qquad \frac{\Gamma\,;[\Delta, A, B]_1 \longrightarrow \gamma}{\Gamma\,;[\Delta, A \otimes B]_1 \longrightarrow \gamma}$$

For weak premisses, we have three additional possibilities:

$$\frac{\Gamma\,;[\Delta, A]_1 \longrightarrow \gamma}{\Gamma\,;[\Delta, A \otimes B]_1 \longrightarrow \gamma} \qquad \frac{\Gamma\,;[\Delta, B]_1 \longrightarrow \gamma}{\Gamma\,;[\Delta, A \otimes B]_1 \longrightarrow \gamma} \qquad \frac{\Gamma\,;[\Delta]_1 \longrightarrow \gamma}{\Gamma\,;[\Delta, A \otimes B]_1 \longrightarrow \gamma}$$

The last of these is actually unnecessary, for $(\Gamma\,;[\Delta]_1 \longrightarrow \gamma) < (\Gamma\,;[\Delta, A \otimes B]_1 \longrightarrow \gamma)$.

In summary, is that there is one weakness agnostic form of the left rule and a pair of special cases to account for weak sequents. This feature is present whenever we have a rule that involves a linear resource in a premiss.

Summary of the formal system The propositions we are now able to support are:

$$A, B, \ldots \quad ::= \quad p \mid A \otimes B \mid \mathbf{1} \mid A \multimap B \mid A \,\&\, B \mid \top \mid A \oplus B \mid \mathbf{0} \mid\, !A$$

As usual in sequents we leave out all hypothesis and judgemental labels unless they are needed to disambiguate. Our syntax for derivations undergoes no change from the previous section, except to extend it in the obvious way for the new connectives.

For the additive rules, the notation $\gamma \cup \gamma'$ should be interpreted as follows:

$$\gamma \cup \gamma' = \begin{cases} C & \text{if } \gamma = \gamma' = C \\ \gamma' & \text{if } \gamma = \cdot \\ \gamma & \text{if } \gamma' = \cdot \end{cases}$$

Note that $\gamma \supseteq \gamma'$ if there exists γ'' such that $\gamma = \gamma' \cup \gamma''$.

Judgemental rules

$$\frac{}{\cdot\,; [p]_0 \longrightarrow p}\ \text{init} \qquad \frac{\Gamma\,; [\Delta, A]_w \longrightarrow \gamma}{\Gamma, A\,; [\Delta]_w \longrightarrow \gamma}\ \text{copy} \qquad \frac{\Gamma, A, A\,; [\Delta]_w \longrightarrow \gamma}{\Gamma, A\,; [\Delta]_w \longrightarrow \gamma}\ \text{factor}$$

Multiplicative rules

$$\frac{\Gamma\,; [\Delta]_w \longrightarrow A \quad \Gamma'\,; [\Delta']_{w'} \longrightarrow B}{\Gamma, \Gamma'\,; [\Delta, \Delta']_{w \vee w'} \longrightarrow A \otimes B}\ \otimes R \qquad \frac{\Gamma\,; [\Delta, A, B]_w \longrightarrow \gamma}{\Gamma\,; [\Delta, A \otimes B]_w \longrightarrow \gamma}\ \otimes L \qquad \frac{\Gamma\,; [\Delta, A_i]_1 \longrightarrow \gamma}{\Gamma\,; [\Delta, A_1 \otimes A_2]_1 \longrightarrow \gamma}\ \otimes L'$$

$$\frac{}{\cdot\,; [\cdot]_0 \longrightarrow \mathbf{1}}\ \mathbf{1}R \qquad \frac{\Gamma\,; [\Delta]_w \longrightarrow \gamma}{\Gamma\,; [\Delta, \mathbf{1}]_w \longrightarrow \gamma}\ \mathbf{1}L$$

$$\frac{\Gamma\,; [\Delta, A]_w \longrightarrow B}{\Gamma\,; [\Delta]_w \longrightarrow A \multimap B}\ \multimap R \qquad \frac{\Gamma\,; [\Delta]_1 \longrightarrow B}{\Gamma\,; [\Delta]_1 \longrightarrow A \multimap B}\ \multimap R$$

$$\frac{\Gamma\,; [\Delta]_w \longrightarrow A \quad \Gamma'\,; [\Delta', B]_{w'} \longrightarrow C}{\Gamma, \Gamma'\,; [\Delta, \Delta', A \multimap B]_{w \vee w'} \longrightarrow C}\ \multimap L$$

Additive rules

$$\frac{\Gamma\,; [\Delta]_w \longrightarrow A \quad \Gamma'\,; [\Delta']_{w'} \longrightarrow B}{\Gamma, \Gamma'\,; [\Delta]_w + [\Delta']_{w'} \longrightarrow A \,\&\, B}\ \&R \qquad \frac{\Gamma\,; [\Delta, A_i]_w \longrightarrow \gamma}{\Gamma\,; [\Delta, A_1 \,\&\, A_2]_w \longrightarrow \gamma}\ \&L_i$$

$$\frac{\Gamma\,; [\Delta]_w \longrightarrow A_i}{\Gamma\,; [\Delta]_w \longrightarrow A_1 \oplus A_2}\ \oplus R_i$$

$$\frac{\Gamma\,; [\Delta, A]_w \longrightarrow \gamma \quad \Gamma'\,; [\Delta']_{w'} \longrightarrow \gamma'}{\Gamma, \Gamma'\,; ([\Delta]_w + [\Delta' \backslash B]_{w'}), A \oplus B \longrightarrow \gamma \cup \gamma'}\ \oplus L \qquad \frac{\Gamma\,; [\Delta]_1 \longrightarrow \gamma \quad \Gamma'\,; [\Delta', B]_{w'} \longrightarrow \gamma'}{\Gamma, \Gamma'\,; ([\Delta \backslash A]_1 + [\Delta']_{w'}), A \oplus B \longrightarrow \gamma \cup \gamma'}\ \oplus L'$$

$$\frac{}{\cdot\,;[\cdot]_1 \longrightarrow \top}\ \top R \qquad \frac{}{\cdot\,;[\mathbf{0}]_1 \longrightarrow \cdot}\ \mathbf{0}L$$

For the $\oplus L$ rules we use the notational convention that $[\Delta]_w, A = [\Delta, A]_w$.

Exponential rules

$$\frac{\Gamma\,;[\cdot]_w \longrightarrow A}{\Gamma\,;[\cdot]_0 \longrightarrow !A}\ !R \qquad \frac{\Gamma\,;[\Delta]_w \longrightarrow \gamma}{\Gamma\backslash A\,;[\Delta, !A]_w \longrightarrow \gamma}\ !L$$

Theorem 3.9 (Soundness of forward derivations)**.**

1. *If $\mathcal{D}_f :: \Gamma\,;[\Delta]_0 \longrightarrow C$, then $(\mathcal{D}_f)^o :: \Gamma\,;\Delta \Longrightarrow C$.*
2. *If $\mathcal{D}_f :: \Gamma\,;[\Delta]_1 \longrightarrow \gamma$, then for any $\Delta' \supseteq \Delta$ and $C \supseteq \gamma$, $(\mathcal{D}_f)^o :: \Gamma\,;\Delta' \Longrightarrow C$.*

Proof. By induction on the structure of $\mathcal{D}_f$. The first important case is "factor", where:

$$\frac{\mathcal{D}'_f :: \Gamma, u : A, v : A\,;[\Delta]_w \longrightarrow \gamma}{\mathcal{D}_f = \text{factor}(u.\ v.\ \mathcal{D}'_f\,; z : A) :: \Gamma, z : A\,;[\Delta]_w \longrightarrow \gamma}\ \text{factor}$$

Case: $w = 0$ and $\gamma = C$.

$(\mathcal{D}'_f)^o :: \Gamma, u : A, v : A\,;\Delta \Longrightarrow C$ — i.h.

$\mathcal{E} = \text{copy}(x.\ \text{id}(x : A)\,; z{:}A) :: \Gamma, z : A\,;\cdot \Longrightarrow A$

$\mathcal{E} +_{v:B} (\mathcal{E} +_{u:A} (\mathcal{D}'_f)^o) :: \Gamma, z : A\,;\Delta \Longrightarrow C$ — theorem 2.21

Case: $w = 1$. Let $\Delta' \supseteq \Delta$ and $C \supseteq \gamma$ be given.

$(\mathcal{D}'_f)^o :: \Gamma, u : A, v : A\,;\Delta' \Longrightarrow C$ — i.h.

$\mathcal{E} = \text{copy}(x.\ \text{id}(x : A)\,; z{:}A) :: \Gamma, z : A\,;\cdot \Longrightarrow A$

$\mathcal{E} +_{v:B} (\mathcal{E} +_{u:A} (\mathcal{D}'_f)^o) :: \Gamma, z : A\,;\Delta' \Longrightarrow C$ — theorem 2.21

For the remaining rule of inference, the induction follows a straightforward pattern. The following is a representative case for a binary right rule.

$$\frac{\mathcal{D}_{f1} :: \Gamma\,;[\Delta]_w \longrightarrow A \quad \mathcal{D}_{f2} :: \Gamma\,;[\Delta']_{w'} \longrightarrow B}{\mathcal{D}_f = \&R(\mathcal{D}_{f1}, \mathcal{D}_{f2}\,; A \,\&\, B) :: \Gamma\,;[\Delta]_w + [\Delta']_{w'} \longrightarrow A \,\&\, B}\ \&R$$

Case: $w = w' = 1$, so $[\Delta]_w + [\Delta']_{w'} = [\Delta \sqcup \Delta']_1$. Let $\Delta'' \supseteq \Delta \sqcup \Delta'$ be given, and note that $\Delta'' \supseteq \Delta$ and $\Delta'' \supseteq \Delta'$.

$$\begin{array}{lr}
(\mathcal{D}_{f1})^o :: \Gamma, \Gamma' \,;\, \Delta'' \Longrightarrow A & \text{i.h. and theorem 2.17} \\
(\mathcal{D}_{f2})^o :: \Gamma, \Gamma' \,;\, \Delta'' \Longrightarrow B & \text{i.h. and theorem 2.17} \\
\&R((\mathcal{D}_{f1})^o, (\mathcal{D}_{f2})^o \,;\, A \,\&\, B) :: \Gamma, \Gamma' \,;\, \Delta'' \Longrightarrow A \,\&\, B & \&R
\end{array}$$

Case: $w = 0$ and $w' = 1$. Then, $[\Delta]_w + [\Delta']_{w'}$ is defined and equals $[\Delta]_0$ if $\Delta' \subseteq \Delta$.

$$\begin{array}{lr}
(\mathcal{D}_{f1})^o :: \Gamma, \Gamma' \,;\, \Delta \Longrightarrow A & \text{i.h. and theorem 2.17} \\
(\mathcal{D}_{f2})^o :: \Gamma, \Gamma' \,;\, \Delta \Longrightarrow B & \text{i.h. and theorem 2.17} \\
\&R((\mathcal{D}_{f1})^o, (\mathcal{D}_{f2})^o \,;\, A \,\&\, B) :: \Gamma, \Gamma' \,;\, \Delta \Longrightarrow A \,\&\, B & \&R
\end{array}$$

Note that in this case the strengthened form of (2) is necessary when using the induction hypothesis on $\mathcal{D}_{f2}$ because weakening the linear context is not valid for backward sequents. The symmetrical case with $w = 1$ and $w' = 0$ follows similarly.

Case: $w = w' = 0$. Then, $[\Delta]_w + [\Delta']_{w'}$ is defined and equals $[\Delta]_0$ if $\Delta = \Delta'$.

$$\begin{array}{lr}
(\mathcal{D}_{f1})^o :: \Gamma, \Gamma' \,;\, \Delta \Longrightarrow A & \text{i.h. and theorem 2.17} \\
(\mathcal{D}_{f2})^o :: \Gamma, \Gamma' \,;\, \Delta \Longrightarrow B & \text{i.h. and theorem 2.17} \\
\&R((\mathcal{D}_{f1})^o, (\mathcal{D}_{f2})^o \,;\, A \,\&\, B) :: \Gamma, \Gamma' \,;\, \Delta \Longrightarrow A \,\&\, B & \&R
\end{array}$$

The cases for the remaining right rules closely follow this pattern. For the left rules, we have one case for the weakness-agnostic forward rule, and several other cases for the rules specific to weak sequents. The following case is representative of the agnostic rules.

$$\frac{\mathcal{D}'_f :: \Gamma \,;\, [\Delta, u : A, v : B]_w \longrightarrow \gamma}{\mathcal{D}_f = \otimes L(u.\, v.\, \mathcal{D}'_f \,;\, z{:}A \otimes B) :: \Gamma \,;\, [\Delta, z : A \otimes B]_w \longrightarrow \gamma}\;\otimes L$$

Case: $w = 0$ and $\gamma = C$.

$$\begin{array}{lr}
(\mathcal{D}'_f)^o :: \Gamma \,;\, \Delta, u : A, v : B \Longrightarrow C & \text{i.h.} \\
\otimes L(u.\, v.\, (\mathcal{D}'_f)^o \,;\, z{:}A \otimes B) :: \Gamma \,;\, \Delta, z{:}A \otimes B \Longrightarrow C & \otimes L
\end{array}$$

Case: $w = 1$. Let $(\Delta', z : A \otimes B) \supseteq (\Delta, z : A \otimes B)$ and $C \supseteq \gamma$ be given.

$$\begin{array}{lr}
(\mathcal{D}'_f)^o :: \Gamma \,;\, \Delta', u : A, v : B \Longrightarrow C & \text{i.h.} \\
\otimes L(u.\, v.\, (\mathcal{D}'_f)^o \,;\, z{:}A \otimes B) :: \Gamma \,;\, \Delta', z{:}A \otimes B \Longrightarrow C & \otimes L
\end{array}$$

Finally, one case that is representative of the rules specific to weak sequents.

$$\frac{\Gamma \,;\, [\Delta, u : A]_1 \longrightarrow \gamma}{\mathcal{D}_f = \otimes L(\mathcal{D}'_f, u, \_ \,;\, v{:}A \otimes B) :: \Gamma \,;\, [\Delta, v : A \otimes B]_1 \longrightarrow \gamma}\;\otimes L'$$

Let $(\Delta', v{:}A \otimes B) \supseteq (\Delta, v{:}A \otimes B)$ and $C \supseteq \gamma$ be given.

$(\mathcal{D}'_f)^o :: \Gamma ; \Delta', u : A, z : B \Longrightarrow C$ i.h.
$\otimes L(u.\, z.\, (\mathcal{D}'_f)^o ; v{:}A \otimes B) :: \Gamma ; \Delta', v{:}A \otimes B \Longrightarrow C$ $\otimes L$

Once again in this case the strengthened form of case (2) is required in order to select a suitable backward sequent. The remaining rules follow one of the above patterns. □

As we saw in theorem 3.3 earlier, the completeness theorem is proved existentially for stronger sequents. We now additionally have to split cases on whether the sequent we infer is weak or strong.

Lemma 3.10. *(Repeated factoring)*
If $\Gamma, \Gamma' ; [\Delta]_w \longrightarrow \gamma$, *then* $\Gamma \sqcup \Gamma' ; [\Delta]_w \longrightarrow \gamma$.

Proof. Repeated applications of "factor". □

Theorem 3.11 (Completeness of forward derivations)**.**
Suppose $\Gamma ; \Delta \Longrightarrow C$. *Then,*

(a) either $\Gamma' ; [\Delta]_0 \longrightarrow C$,
(b) or $\Gamma' ; [\Delta']_1 \longrightarrow \gamma$

for some $\Gamma' \subseteq \Gamma$, $\Delta' \subseteq \Delta$ *and* $\gamma \subseteq C$.

Proof. Induction on the structure of the derivation of $\mathcal{D}_b :: \Gamma ; \Delta \Longrightarrow C$. The following are some representative cases.

Case: initial sequents, i.e., $\mathcal{D}_b = \mathrm{init}(u{:}p) :: \Gamma ; u : p \Longrightarrow p$.
In this case, $\mathrm{init}(u{:}p) :: \cdot ; [u : p]_0 \longrightarrow p$.

Case: "copy", i.e., $\mathcal{D}_b = \mathrm{copy}(\mathcal{D}'_b, u ; v{:}A) :: \Gamma, u : A ; \Delta \Longrightarrow C$, i.e., $\mathcal{D}'_b :: \Gamma, u : A ; \Delta, v : A \Longrightarrow C$. Here there are three possibilities for the induction hypothesis on $\mathcal{D}'_b$:

Subcase: $\Gamma' ; [\Delta, v : A]_0 \longrightarrow C$ for some $\Gamma' \subseteq \Gamma, u : A$.
Then, by "copy", $\Gamma', u' : A ; [\Delta]_0 \longrightarrow C$. If $u \in \mathrm{dom}(\Gamma)$, then use combine u and u' by means of "factor"; otherwise, set u' to u.

Subcase: $\Gamma' ; [\Delta', v : A]_1 \longrightarrow \gamma$ for some $\Gamma' \subseteq (\Gamma, u : A)$, $(\Delta', v : A) \subseteq \Delta$, and $\gamma \subseteq C$. A similar argument as the previous case applies here, except we obtain $\Gamma', u : A ; [\Delta']_1 \longrightarrow \gamma$.

Subcase: $\Gamma' ; [\Delta']_1 \longrightarrow \gamma$ for some $\Gamma' \subseteq (\Gamma, u : A)$, $(\Delta') \subseteq \Delta$, and $\gamma \subseteq C$ and $v \notin \mathrm{dom}(\Delta')$. In this case, the sequent itself satisfies case (b).

Case: a multiplicative rule such as:

$$\frac{\mathcal{D}_{b1} :: \Gamma \mathrel{;} \Delta_1 \Longrightarrow A \quad \mathcal{D}_{b2} :: \Gamma \mathrel{;} \Delta_2 \Longrightarrow B}{\mathcal{D}_b = \otimes R(\mathcal{D}_{b1}, \mathcal{D}_{b2} \mathrel{;} A \otimes B) :: \Gamma \mathrel{;} \Delta_1, \Delta_2 \Longrightarrow A \otimes B}\ \otimes R$$

We obtain the following cases for the induction hypothesis on $\mathcal{D}_{b1}$ and $\mathcal{D}_{b2}$.

Subcase: $\Gamma_1 \mathrel{;} [\Delta_1]_0 \longrightarrow A$ and $\Gamma_2 \mathrel{;} [\Delta_2]_0 \longrightarrow B$ for some $\Gamma_1 \subseteq \Gamma$ and $\Gamma_2 \subseteq \Gamma$. By $\otimes R$ we have $\Gamma_1, \Gamma_2 \mathrel{;} [\Delta_1, \Delta_2]_0 \longrightarrow A \otimes B$, then we appeal to lemma 3.10 and note that $\Gamma_1 \sqcup \Gamma_2 \subseteq \Gamma$.

Subcase: $\Gamma_1 \mathrel{;} [\Delta_1']_1 \longrightarrow \gamma$ and $\Gamma_2 \mathrel{;} [\Delta_2]_0 \longrightarrow B$ for some $\Gamma_1 \subseteq \Gamma$, $\Gamma_2 \subseteq \Gamma$, $\Delta_1' \subseteq \Delta_1$ and $\gamma \subseteq A$. If $\gamma = \cdot$, then the first sequent already satisfies case (b), so we only need to consider $\gamma = A$. In this case, from $\otimes R$, we get $\Gamma_1, \Gamma_2 \mathrel{;} [\Delta_1', \Delta_2]_1 \longrightarrow A \otimes B$, after which we use the same argument as the previous case.

Subcase: $\Gamma_1 \mathrel{;} [\Delta_1']_1 \longrightarrow \gamma_A$ and $\Gamma_2 \mathrel{;} [\Delta_2']_1 \longrightarrow \gamma_B$ for some $\Gamma_1 \subseteq \Gamma$, $\Gamma_2 \subseteq \Gamma$, $\Delta_1' \subseteq \Delta_1$, $\Delta_2' \subseteq \Delta_2$, $\gamma_A \subseteq A$ and $\gamma_B \subseteq B$. Like the previous case, we only need to consider the cases for which $\gamma_A = A$ and $\gamma_B = B$; then, by $\otimes R$ we get $\Gamma_1, \Gamma_2 \mathrel{;} [\Delta_1', \Delta_2']_1 \longrightarrow A \otimes B$, and use the same argument as before.

Case: an additive rule, such as:

$$\frac{\mathcal{D}_{b1} :: \Gamma \mathrel{;} \Delta \Longrightarrow A \quad \mathcal{D}_{b2} :: \Gamma \mathrel{;} \Delta \Longrightarrow B}{\mathcal{D}_b = \&R(\mathcal{D}_{b1}, \mathcal{D}_{b2} \mathrel{;} A \otimes B) :: \Gamma \mathrel{;} \Delta \Longrightarrow A \mathbin{\&} B}\ \&R$$

We obtain the following cases for the induction hypothesis on $\mathcal{D}_{b1}$ and $\mathcal{D}_{b2}$.

Subcase: $\Gamma_1 \mathrel{;} [\Delta]_0 \longrightarrow A$ and $\Gamma_2 \mathrel{;} [\Delta]_0 \longrightarrow B$ for some $\Gamma_1 \subseteq \Gamma$ and $\Gamma_2 \subseteq \Gamma$. By $\&R$, we obtain $\Gamma_1, \Gamma_2 \mathrel{;} [\Delta]_0 \Longrightarrow A \mathbin{\&} B$, and we appeal to lemma 3.10 as before.

Subcase: $\Gamma_1 \mathrel{;} [\Delta']_1 \longrightarrow \gamma$ and $\Gamma_2 \mathrel{;} [\Delta]_0 \longrightarrow B$ for some $\Gamma_1 \subseteq \Gamma$, $\Gamma_2 \subseteq \Gamma$, $\Delta' \subseteq \Delta$ and $\gamma \subseteq A$. As before, the only interesting case is with $\gamma = A$, in which case we obtain by $\&R$ and definition 3.8 that $\Gamma_1, \Gamma_2 \mathrel{;} [\Delta]_0 \longrightarrow A \mathbin{\&} B$, after which we use the same argument as before.

Subcase: $\Gamma_1 \mathrel{;} [\Delta_1']_1 \longrightarrow \gamma_A$ and $\Gamma_2 \mathrel{;} [\Delta_2']_1 \longrightarrow \gamma_B$ for some $\Gamma_1 \subseteq \Gamma$, $\Gamma_2 \subseteq \Gamma$, $\Delta_1' \subseteq \Delta$, $\Delta_2' \subseteq \Delta$, $\gamma_A \subseteq A$ and $\gamma_B \subseteq B$. Again the interesting case is when $\gamma_A = A$ and $\gamma_B = B$, whereupon by $\&R$ we get $\Gamma_1, \Gamma_2 \mathrel{;} [\Delta_1 \sqcup \Delta_2]_1 \longrightarrow A \mathbin{\&} B$ and we note that $\Delta_1 \sqcup \Delta_2 \subseteq \Delta$.

The cases for the exponential rules are very straightforward. □

3.3 Optimization: affine resources

In the absence of negative **1** in the logic, the forward calculus of the previous section suffices to remove all resource non-determinism. With the addition of **1**, particularly negative occurrences, we have a problem of *affine non-determinism*, which arises from the interaction of $\mathbf{1}$ with other connectives. For most connectives, $\mathbf{1}$ has only a *unitary* function, where an equivalent proposition can be found which does not require (that particular instance of) $\mathbf{1}$. By considering all instances of $\mathbf{1}$ appearing as an operand in a connective, the full list of such equivalences is as follows:[2]

$$A \otimes \mathbf{1} \equiv A \equiv \mathbf{1} \otimes A \qquad \mathbf{1} \multimap A \equiv A \qquad \mathbf{1} \mathbin{\&} \mathbf{1} \equiv \mathbf{1} \qquad \mathbf{1} \oplus \mathbf{1} \equiv \mathbf{1} \qquad !\mathbf{1} \equiv \mathbf{1}$$

For the rest of this paper we assume a logic in $\mathbf{1}$-normal form ($\mathbf{1}$nf), which we define as that fragment without unitary uses of $\mathbf{1}$. This simpler fragment allows an examination of the occurrences of $\mathbf{1}$ actually relevant to resource management. Unless specified, we assume for the rest of this section that all propositions are in $\mathbf{1}$nf.

Definition 3.12 ($\mathbf{1}$ normal form)**.** *Given a proposition A, its $\mathbf{1}$ normal form, $(A)_{\mathbf{1}}$ is defined as follows.*

$$(A \otimes B)_{\mathbf{1}} = \begin{cases} (A)_{\mathbf{1}} & \text{if } (B)_1 = \mathbf{1} \\ (B)_{\mathbf{1}} & \text{if } (A)_1 = \mathbf{1} \\ (A)_{\mathbf{1}} \otimes (B)_{\mathbf{1}} & \text{otherwise} \end{cases} \qquad (\mathbf{1})_{\mathbf{1}} = \mathbf{1}$$

$$(A \multimap B)_{\mathbf{1}} = \begin{cases} (B)_{\mathbf{1}} & \text{if } (A)_{\mathbf{1}} = \mathbf{1} \\ (A)_{\mathbf{1}} \multimap (B)_{\mathbf{1}} & \text{otherwise} \end{cases}$$

$$(A \mathbin{\&} B)_{\mathbf{1}} = \begin{cases} \mathbf{1} & \text{if } (A)_{\mathbf{1}} = (B)_{\mathbf{1}} = \mathbf{1} \\ (A)_{\mathbf{1}} \mathbin{\&} (B)_{\mathbf{1}} & \text{otherwise} \end{cases} \qquad (\top)_{\mathbf{1}} = \top$$

$$(A \oplus B)_{\mathbf{1}} = \begin{cases} \mathbf{1} & \text{if } (A)_{\mathbf{1}} = (B)_{\mathbf{1}} = \mathbf{1} \\ (A)_{\mathbf{1}} \oplus (B)_{\mathbf{1}} & \text{otherwise} \end{cases} \qquad (\mathbf{0})_{\mathbf{1}} = \mathbf{0}$$

$$(!A)_{\mathbf{1}} = \begin{cases} \mathbf{1} & \text{if } (A)_{\mathbf{1}} = \mathbf{1} \\ !(A)_{\mathbf{1}} & \text{otherwise} \end{cases}$$

An interesting class of propositions has the form $A \mathbin{\&} \mathbf{1}$ or $\mathbf{1} \mathbin{\&} A$; as a resource, $A \mathbin{\&} \mathbf{1}$ provides a choice of either using A linearly in the proof, or not using A at all, *i.e.*, it encodes

[2]In the presence of quantifiers, we have some additional equivalences: $\forall x.\mathbf{1} \equiv \mathbf{1}$ and $\exists x.\mathbf{1} \equiv \mathbf{1}$

an *at-most one use* or *affine* interpretation. Indeed, such propositions allow us to recover affine logic in the exact setting of linear logic, by translating affine implications $A \rightarrow B$ into $A \mathbin{\&} \mathbf{1} \multimap B$. There is another, more popular embedding of affine logic into linear logic that translates $A \rightarrow B$ into $A \multimap B \otimes \top$. The difference between the two encodings manifests as a choice between a *local* and a *global* translation — translating into $A \mathbin{\&} \mathbf{1} \multimap B$ doesn't destroy the linear nature of resources, but $A \multimap B \otimes \top$ makes every resource affine because of the presence of positive $\otimes\top$. Yet, and despite the fact that positive $\top$ complicates backwards search, encodings in logic programming languages like Lolli use the second encoding because & is disallowed in the body of clauses. Rather than repeat this approach and disallow $\mathbf{1}$ in the syntax where it is problematic, for the rest of this section we examine the nature of resource non-determinism caused by such instances $\mathbf{1}$.

Characterising non-unitary uses of 1 First, consider the effect of removing $\mathbf{1}L$ entirely from the logic. In the $\mathbf{1}$nf fragment, only the following instances of $\mathbf{1}$ remain: $A \multimap \mathbf{1}$, $A \mathbin{\&} \mathbf{1}$, $\mathbf{1} \mathbin{\&} A$, $A \oplus \mathbf{1}$, $\mathbf{1} \oplus A$ and the formula $\mathbf{1}$ itself. On the right, the corresponding rules are fully deterministic. On the left, all of these forms – except $\mathbf{1}$ itself – have the following specialized rules:

$$\frac{\Gamma ; \Delta \Longrightarrow A \quad \Gamma ; \Delta' \Longrightarrow C}{\Gamma ; \Delta, \Delta', A \multimap \mathbf{1} \Longrightarrow C} \multimap\!\mathbf{1}L$$

$$\frac{\Gamma ; \Delta \Longrightarrow C}{\Gamma ; \Delta, A \mathbin{\&} \mathbf{1} \Longrightarrow C} \&\mathbf{1}L_1 \qquad \frac{\Gamma ; \Delta, A \Longrightarrow C}{\Gamma ; \Delta, A \mathbin{\&} \mathbf{1} \Longrightarrow C} \&\mathbf{1}L_2$$

$$\frac{\Gamma ; \Delta \Longrightarrow C}{\Gamma ; \Delta, \mathbf{1} \mathbin{\&} A \Longrightarrow C} \mathbf{1}\&L_1 \qquad \frac{\Gamma ; \Delta, A \Longrightarrow C}{\Gamma ; \Delta, \mathbf{1} \mathbin{\&} A \Longrightarrow C} \mathbf{1}\&L_2$$

$$\frac{\Gamma ; \Delta, A \Longrightarrow C \quad \Gamma ; \Delta \Longrightarrow C}{\Gamma ; \Delta, A \oplus \mathbf{1} \Longrightarrow C} \oplus\mathbf{1}L \qquad \frac{\Gamma ; \Delta \Longrightarrow C \quad \Gamma ; \Delta, A \Longrightarrow C}{\Gamma ; \Delta, \mathbf{1} \oplus A \Longrightarrow C} \mathbf{1}\oplus L$$

We have described the situation with $\&\mathbf{1}$ and $\mathbf{1}\&$ before and clearly visible above in the pair of rules $\mathbf{1}\&L_1$ and $\&\mathbf{1}L_1$, formulas of the form $A \mathbin{\&} \mathbf{1}$ define an affine interpretation for the resource A. We examine this case in detail in the next section. For $\mathbf{1}\oplus L$ and $\oplus\mathbf{1}L$, the premisses appear to give the formula A a meaning of optional use – we can prove the conclusion C both in the presence and absence of A. In fact, one might view this kind of optional use (one *and* zero times) as the external version of the affine case (at-most one use); thus, one might imagine a substructural logic where external options are internalised using locally sound and complete introduction/elimination rules. Fortunately, the treatment of the affine case in the next section provides a satisfactory answer for the optional case also.

For the $\multimap \mathbf{1}L$ rule, we do not have a satisfactory treatment. In fact, we can certainly construct examples where this rule can be iterated indefinitely, giving larger and large sequents.

$$\dfrac{\Gamma ; \Delta \Longrightarrow A \qquad \dfrac{\Gamma ; \Delta \Longrightarrow A \qquad \dfrac{\Gamma ; \Delta \Longrightarrow A \quad \Gamma ; \Delta', \mathbf{1} \Longrightarrow C}{\Gamma ; \Delta, \Delta', A \multimap \mathbf{1} \Longrightarrow C}}{\Gamma ; \Delta, \Delta, \Delta', A \multimap \mathbf{1}, A \multimap \mathbf{1} \Longrightarrow C}}{\vdots}$$

We leave a treatment of this and other sources of unknown use non-determinism to future work, but note that that no complete solution can exist because of the undecidability of multiplicative-additive-exponential linear logic. On the other hand, categorizing and solving other kinds of unknown use non-determinism can give decision procedures for larger fragments. These investigations will depend on the need for the increased expressivity; for example, by showing how a negative $A \multimap \mathbf{1}$ gives a more natural encoding than other possibilities.

In the next section we give first a backward and then a forward calculus to handle the affine case. One particular note – we remove all hypotheses $\mathbf{1}$ in the ultimate goal sequent. Thus, we never need to use the $\mathbf{1}L$ rule at all, so we just discard it. We can easily add these extra $\mathbf{1}$s to the goal sequent if needed after search completes.

3.3.1 Affine zones for the backward calculus

To handle the affine resources, we insert a new *affine context* Ψ among the hypotheses of sequents, giving the following shape for sequents: $\Gamma ; \Psi ; \Delta \Longrightarrow C$. We view this affine zone as a multiset of formulas, just like the linear zone, but with an additional structurally admissible rule of weakening (theorem 3.13). The resulting logic can be seen as a fragment of Hodas' Omnibus logic[54], which has a strict zone in addition to the affine context.

For the judgemental rules, we have a rule of *promotion* to turn an affine hypothesis into a linear hypothesis. This corresponds to committing to an actual use of the affine resource.

$$\frac{\Gamma ; \Psi ; \Delta, A \Longrightarrow C}{\Gamma ; \Psi, A ; \Delta \Longrightarrow C}\ \text{promote}$$

On the other hand, affine resources can remain unused because we allow any number of them to escape through initial and other axiomatic sequents:

$$\frac{}{\Gamma\,;\Psi\,;A \Longrightarrow A}\text{ init} \qquad \frac{}{\Gamma\,;\Psi\,;\cdot \Longrightarrow \mathbf{1}}\ \mathbf{1}R \qquad \frac{}{\Gamma\,;\Psi\,;\Delta \Longrightarrow \top}\ \top R \qquad \frac{}{\Gamma\,;\Psi\,;\Delta,\mathbf{0} \Longrightarrow C}\ \mathbf{0}L$$

We distinguish propositions of the form $A \,\&\, \mathbf{1}$ and $\mathbf{1} \,\&\, A$ by treating $\&\mathbf{1}$, $\mathbf{1}\&$, $\mathbf{1}\oplus$, etc. as operators. However, unlike the internalisation of judgemental rules as connectives, these operators are not true connectives as they only occur on the left of sequents, and use the usual right rules for & and **1** to infer $A \,\&\, \mathbf{1}$ on the right. Note that this is a departure from the judgemental presentation laid out in chapter 2.1; however, it can be formalised fully in this style if we make use of another auxiliary judgement to represent **1** on the right. We do not take this step because the calculus with affine zones is not being put forward as an extension of the linear sequent calculus with its own independent interest, but rather to justify an improvement in the treatment of affine resources in the forward calculus.

Now for the rules of this calculus. All the sequents are assumed to have propositions in **1**nf. No rules require **1** as a resource, but we have (derived) rules for the situations where **1** occurs as an operand of the principal connective. To enforce an absence of one among the hypotheses, we add some side-conditions to &L.

Judgemental Rules

$$\frac{}{\Gamma\,;\Psi\,;p \Longrightarrow p}\text{ init} \qquad \frac{\Gamma,A\,;\Psi\,;\Delta,A \Longrightarrow C}{\Gamma,A\,;\Psi\,;\Delta \Longrightarrow C}\text{ copy} \qquad \frac{\Gamma\,;\Psi\,;\Delta,A \Longrightarrow C}{\Gamma\,;\Psi,A\,;\Delta \Longrightarrow C}\text{ promote}$$

Multiplicative rules

$$\frac{\Gamma\,;\Psi\,;\Delta \Longrightarrow A \quad \Gamma\,;\Psi'\,;\Delta' \Longrightarrow B}{\Gamma\,;\Psi,\Psi'\,;\Delta,\Delta' \Longrightarrow A \otimes B}\ \otimes R \qquad \frac{\Gamma\,;\Psi\,;\Delta,A,B \Longrightarrow C}{\Gamma\,;\Psi\,;\Delta,A \otimes B \Longrightarrow C}\ \otimes L$$

$$\frac{}{\Gamma\,;\Psi\,;\cdot \Longrightarrow \mathbf{1}}\ \mathbf{1}R \qquad \text{no } \mathbf{1}L$$

$$\frac{\Gamma\,;\Psi\,;\Delta,A \Longrightarrow B}{\Gamma\,;\Psi\,;\Delta \Longrightarrow A \multimap B}\ \multimap R$$

$$\frac{\Gamma\,;\Psi\,;\Delta \Longrightarrow A \quad \Gamma\,;\Psi'\,;\Delta,B \Longrightarrow C \quad B \neq \mathbf{1}}{\Gamma\,;\Psi,\Psi'\,;\Delta,\Delta',A \multimap B \Longrightarrow C}\ \multimap L \qquad \frac{\Gamma\,;\Psi\,;\Delta \Longrightarrow A \quad \Gamma\,;\Psi'\,;\Delta \Longrightarrow C}{\Gamma\,;\Psi,\Psi'\,;\Delta,\Delta',A \multimap \mathbf{1} \Longrightarrow C}\ \mathbf{1}{\multimap}L$$

Additive rules

$$\frac{\Gamma\,;\Psi\,;\Delta \Longrightarrow A \quad \Gamma\,;\Psi\,;\Delta \Longrightarrow B}{\Gamma\,;\Psi\,;\Delta \Longrightarrow A \,\&\, B}\ \&R \qquad \frac{}{\Gamma\,;\Psi\,;\Delta \Longrightarrow \top}\ \top R$$

$$\frac{\Gamma\,;\Psi\,;\Delta, A \Longrightarrow C \quad B \neq \mathbf{1}}{\Gamma\,;\Psi\,;\Delta, A \,\&\, B \Longrightarrow C}\ \&L_1 \qquad \frac{\Gamma\,;\Psi\,;\Delta, B \Longrightarrow C \quad A \neq \mathbf{1}}{\Gamma\,;\Psi\,;\Delta, A \,\&\, B \Longrightarrow C}\ \&L_2$$

$$\frac{\Gamma\,;\Psi, A\,;\Delta \Longrightarrow C}{\Gamma\,;\Psi\,;\Delta, A \,\&\, \mathbf{1} \Longrightarrow C}\ \&\mathbf{1}L \qquad \frac{\Gamma\,;\Psi, B\,;\Delta \Longrightarrow C}{\Gamma\,;\Psi\,;\Delta, \mathbf{1} \,\&\, B \Longrightarrow C}\ \mathbf{1}\&L$$

$$\frac{\Gamma\,;\Psi\,;\Delta \Longrightarrow A}{\Gamma\,;\Psi\,;\Delta \Longrightarrow A \oplus B}\ \oplus R_1 \qquad \frac{\Gamma\,;\Psi\,;\Delta \Longrightarrow B}{\Gamma\,;\Psi\,;\Delta \Longrightarrow A \oplus B}\ \oplus R_2$$

$$\frac{\Gamma\,;\Psi\,;\Delta, A \Longrightarrow C \quad \Gamma\,;\Psi\,;\Delta, B \Longrightarrow C \quad A \neq \mathbf{1} \quad B \neq \mathbf{1}}{\Gamma\,;\Psi\,;\Delta, A \oplus B \Longrightarrow C}\ \oplus L \qquad \frac{}{\Gamma\,;\Psi\,;\Delta, \mathbf{0} \Longrightarrow C}\ \mathbf{0}L$$

$$\frac{\Gamma\,;\Psi\,;\Delta, A \Longrightarrow C \quad \Gamma\,;\Psi\,;\Delta \Longrightarrow C \quad A \neq \mathbf{1}}{\Gamma\,;\Psi\,;\Delta, A \oplus \mathbf{1} \Longrightarrow C}\ \oplus\mathbf{1}L \qquad \frac{\Gamma\,;\Psi\,;\Delta \Longrightarrow C \quad \Gamma\,;\Psi\,;\Delta, B \Longrightarrow C \quad B \neq \mathbf{1}}{\Gamma\,;\Psi\,;\Delta, \mathbf{1} \oplus B \Longrightarrow C}\ \mathbf{1}\oplus L$$

Exponential rules

$$\frac{\Gamma, A\,;\Psi\,;\Delta \Longrightarrow C}{\Gamma\,;\Psi\,;\Delta, !A \Longrightarrow C}\ !L \qquad \frac{\Gamma\,;\cdot\,;\cdot \Longrightarrow A}{\Gamma\,;\Psi\,;\cdot \Longrightarrow !A}\ !R$$

This presentation of a resource-management motivated three zoned logic bears a strong resemblance to a similar system of Cervesato *et al.* [23] for the domain of (backward-reasoning) linear logic programming in the uniform fragment. The primary difference lies in the interpretation of the new zone – *strict* in [23] versus affine in this work. The design of their three-zoned system derives its primary motivation from the nature of & and ⊤, with the strict contexts designed to handle the additive nature of &. In a similar sense in which strict contexts arise for a systematic approach to resource management in backward search, we claim that affine contexts arise naturally in the setting of forward search.

Structural properties. We obtain an easily shown admissible structural weakening theorem for the affine context, in addition to the straightforward extension of the structural properties for the unrestricted context in theorem 2.17 to the three-zoned setting. Contraction, of course, is not admissible for the affine context, though it continues to be admissible for the unrestricted context.

Theorem 3.13 (Structural properties)**.**

1. *If $\mathcal{D} :: \Gamma\,;\Psi\,;\Delta \Longrightarrow C$, then $\mathcal{D} :: \Gamma, \Gamma'\,;\Psi, \Psi'\,;\Delta \Longrightarrow C$ for any Γ' and Ψ'. (Weakening)*
2. *If $\mathcal{D} :: \Gamma, u : A, v : A\,;\Psi\,;\Delta \Longrightarrow C$, then $[v/u]\mathcal{D} :: \Gamma, v{:}A\,;\Psi\,;\Delta \Longrightarrow C$. (Contraction)*

Proof. Induction on the structure of $\mathcal{D}$. □

We also add two new cases to the cut rule to cut the affine resources:

$$\frac{\Gamma\,;\Psi\,;\cdot \Longrightarrow A \quad \Gamma\,;\Psi', A\,;\Delta' \Longrightarrow C}{\Gamma\,;\Psi, \Psi'\,;\Delta' \Longrightarrow C}\ \text{affine cut} \qquad \frac{\Gamma\,;\Psi\,;\Delta \Longrightarrow \mathbf{1} \quad \Gamma\,;\Psi'\,;\Delta' \Longrightarrow C}{\Gamma\,;\Psi, \Psi'\,;\Delta, \Delta' \Longrightarrow C}\ \mathbf{1}\ \text{cut}$$

The first case is a straightforward statement that a goal may be used once. The second case, however, is unusual for cuts in the judgemental philosophy because the cut proposition, **1**, is not required to occur in the second premiss at all. This case corresponds to having **1** as a hypothesis in the dyadic system of chapter 2.1 without affine resources. Alternatively, one might view a sequent $\Gamma\,;\Psi\,;\Delta \Longrightarrow \mathbf{1}$ as an internalisation of a new judgemental form $\Gamma\,;\Psi\,;\Delta \Longrightarrow \#$, with # defined as a condition where all linear resources are consumed. This form of cut is familiar from Girard's MIX rule for classical two-sided sequent calculi:

$$\frac{\Gamma \Longrightarrow \Delta \quad \Gamma' \Longrightarrow \Delta'}{\Gamma, \Gamma' \Longrightarrow \Delta, \Delta'}\ \text{MIX}$$

This suggestive similarity can be formalized in great detail by translating from classical to intuitionistic linear logic. A more complete exposition is given in Chang *et al.* [27], where the logic with MIX rules is shown to be equivalent to a logic of resource consumption, which in turn gives a judgemental explanation for MIX.

Theorem 3.14 (Admissibility of cut)**.**

(1) If $\Gamma\,;\Psi\,;\Delta \Longrightarrow \mathbf{1}$ and $\Gamma\,;\Psi'\,;\Delta' \Longrightarrow C$, then $\Gamma\,;\Psi, \Psi'\,;\Delta, \Delta' \Longrightarrow C$.
(2) If $\Gamma\,;\Psi\,;\Delta \Longrightarrow A$ and $\Gamma\,;\Psi'\,;\Delta', A \Longrightarrow C$, then $\Gamma\,;\Psi, \Psi'\,;\Delta, \Delta' \Longrightarrow C$.
(3) If $\Gamma\,;\Psi\,;\cdot \Longrightarrow A$ and $\Gamma\,;\Psi', A\,;\Delta \Longrightarrow C$, then $\Gamma\,;\Psi, \Psi'\,;\Delta \Longrightarrow C$.
(4) If $\Gamma\,;\cdot\,;\cdot \Longrightarrow A$ and $\Gamma, A\,;\Psi\,;\Delta \Longrightarrow C$, then $\Gamma\,;\Psi\,;\Delta \Longrightarrow C$.

Proof sketch. The proof is a straightforward extension of that of theorem 2.21. We shall omit the details of the constructive formalisation of the cut elimination procedure, and instead sketch just the major differences from theorem 2.21. As usual, we name the three derivations $\mathcal{D}$, $\mathcal{E}$ and $\mathcal{F}$. The lexicographic ordering is extended slightly by allowing proofs of kind (3) to be used in those of kinds (2) and (1); and those of kind (4) to be used for kinds (3), (2) and (1). □

Correctness. While cut gives us global soundness of the sequent calculus with affine resources, we still need to prove that it is sound and complete with respect to the dyadic system. In order to show soundness, we employ a useful shorthand, $\Psi \mathbin{\&} \mathbf{1}$, to stand for a context consisting of every proposition A in Ψ replaced with $A \mathbin{\&} \mathbf{1}$ or $\mathbf{1} \mathbin{\&} A$. We obtain a succinct soundness theorem.

Lemma 3.15. *If* $\Gamma\,;\Psi \mathbin{\&} \mathbf{1}, \Delta \Longrightarrow C$*, then* $\Gamma\,; (\Psi, \Psi') \mathbin{\&} \mathbf{1}, \Delta \Longrightarrow C$*.*

Proof. Chain a sequence of $\mathbf{1}L$ and $\&L$ rules. □

Theorem 3.16 (Soundness)**.** *If* $\Gamma\,;\Psi\,;\Delta \Longrightarrow C$*, then* $\Gamma\,;\Psi \mathbin{\&} \mathbf{1}, \Delta \Longrightarrow C$*.*

Proof. By induction on the structure of the derivation $\mathcal{D} :: \Gamma\,;\Psi\,;\Delta \Longrightarrow C$. All cases except "init" and $\mathbf{1}R$ have trivial verifications. For these two rules, we appeal to lemma 3.15. □

We also obtain a strong completeness theorem, schematic for affine contexts.

Theorem 3.17 (Completeness)**.** *If* $\Gamma\,;\Delta \Longrightarrow C$*, then* $\Gamma\,;\Psi\,;\Delta \Longrightarrow C$ *for any* Ψ*.*

Proof. By straightforward structural induction on the derivation of $\Gamma\,;\Delta \Longrightarrow C$. □

This section has served primarily a motivational purpose; we now turn our attention to our original goal of controlling affine non-determinism in forward reasoning.

3.3.2 Affine contexts in the forward calculus

Like before with the unrestricted contexts, in the forward direction we create only that subset of the affine context that we can infer from other premisses and the conclusion, with the sole difference that in order to maintain the affine interpretation, we treat the affine context multiplicatively. Rules for formulas with $\mathbf{1}$ as an operand require particular attention; for example, consider the following tempting possibilities for $A \multimap \mathbf{1}$:

$$\frac{\Gamma\,;\Psi, A\,;[\Delta]_w \longrightarrow \gamma}{\Gamma\,;\Psi\,;[\Delta, A \mathbin{\&} \mathbf{1}]_w \longrightarrow \gamma}\ \&\mathbf{1}L \qquad \frac{\Gamma\,;\Psi\,;[\Delta]_w \longrightarrow \gamma}{\Gamma\,;\Psi\,;[\Delta, A \mathbin{\&} \mathbf{1}]_w \longrightarrow \gamma}\ \&\mathbf{1}L'$$

The $\&\mathbf{1}L'$ rule lacks any structural control on the number of occurrences of $A \mathbin{\&} \mathbf{1}$. We have already seen this problem before in the presence of the $\mathbf{1}L$ rule, removing which makes the iterative nature of this rule obvious. We attack this problem by treating this second instance as a kind of weakening; thus, we use the second of the above rules only after we have more information about the multiplicity of $A \mathbin{\&} \mathbf{1}$.

When do we learn anything about the multiplicity of a formula? Certainly, we can never infer the exact multiplicity of any given formula by just looking at the final goal sequent – this would make the fragment decidable, and we already know that linear logic in the presence of additive connectives is undecidable. However, we do know that the multiplicity of linear $A \mathbin{\&} \mathbf{1}$ exceeds the multiplicity of A in the affine context; this suffices to control the iteration of $\&\mathbf{1}L'$ as follows – remove this rule entirely from consideration during search, and assume for every other rule with a weak premiss that the formula $A \mathbin{\&} \mathbf{1}$ exists implicitly in the linear context.

Of course, in the proof theory it becomes tedious to modify every logical rule with the tests and side conditions corresponding to these implicitly present affine resources, so we introduce a layer of abstraction between the inference rule and the matching conditions that enable the rule. Conceptually, matching conditions in the forward direction generalize the notion of *occurrence* in a context, written exactly like adjunctions (Γ, A) for historical reasons. This notation makes perfect sense in backward reasoning, because the contexts, ambiently or explicitly, serve as parameters for the search procedure. In contrast, because information flows in the opposite direction in forward reasoning, inference rules construct the contexts of the conclusion from those of the premisses, treating contexts as localized (first-class) objects.[3] As a matching condition, adjunction describes only the rather simple condition of occurrence.

In order to obtain a more complex and process-oriented view of matching, we define a new judgement on zoned contexts $\Gamma \,;\, \Psi \,;\, \Delta$ (written Υ):

$$\Upsilon \vDash \Upsilon' + \Delta'$$

which we read "*Υ admits the decomposition Υ', Δ'.*" We abuse notation slightly to write $[\Upsilon]_w$ to stand for $\Gamma \,;\, \Psi \,;\, [\Delta]_w$ if $\Upsilon = \Gamma \,;\, \Psi \,;\, \Delta$. The modes for this judgement are somewhat subtle: it takes $[\Upsilon]_w$ and Δ as input, and produces the output $[\Upsilon']$ if it succeeds. The rules

[3]We find this phenomenon in an even stronger form when we add quantifiers and relax all equalities to unifiability – existential variables in backward search are treated *globally*, affecting otherwise disjoint branches in the derivation tree, and requiring undo operations for backtracking. Forward reasoning localizes these variables, giving a much simpler view of unification.

for this judgement proceed purely in the bottom-up direction, with the output Υ' read off from the completed derivation. The simplest rule for this judgement merely admits the trivial adjunction.

$$\frac{}{\Upsilon \vDash \Upsilon + \cdot} \vDash_{=}$$

The remaining rules fall into three categories for the three different zones. For the linear zone:

$$\frac{\Gamma \,;\, \Psi \,;\, \Delta \vDash \Upsilon + \Delta'}{\Gamma \,;\, \Psi \,;\, \Delta, A \vDash \Upsilon + \Delta', A} \vDash_{\text{linear}}$$

For the affine zone:

$$\frac{\Gamma \,;\, \Psi \,;\, \Delta \vDash \Upsilon + \Delta'}{\Gamma \,;\, \Psi, A \,;\, \Delta \vDash \Upsilon + \Delta', A \mathbin{\&} \mathbf{1}} \vDash_{\&\mathbf{1}} \qquad \frac{\Gamma \,;\, \Psi \,;\, \Delta \vDash \Upsilon + \Delta'}{\Gamma \,;\, \Psi, A \,;\, \Delta \vDash \Upsilon + \Delta', \mathbf{1} \mathbin{\&} A} \vDash_{\mathbf{1}\&}$$

$$\frac{\Gamma \,;\, \Psi \,;\, \Delta \vDash \Upsilon + \Delta'}{\Gamma \,;\, \Psi \,;\, \Delta \vDash \Upsilon + \Delta', A \mathbin{\&} \mathbf{1}} \vDash'_{\&\mathbf{1}} \qquad \frac{\Gamma \,;\, \Psi \,;\, \Delta \vDash \Upsilon + \Delta'}{\Gamma \,;\, \Psi \,;\, \Delta \vDash \Upsilon + \Delta', \mathbf{1} \mathbin{\&} A} \vDash'_{\mathbf{1}\&}$$

For the unrestricted zone:

$$\frac{\Gamma \,;\, \Psi \,;\, \Delta \vDash \Upsilon + \Delta'}{\Gamma, A \,;\, \Psi \,;\, \Delta \vDash \Upsilon + \Delta', !A} \vDash_{!} \qquad \frac{\Gamma \,;\, \Psi \,;\, \Delta \vDash \Upsilon + \Delta'}{\Gamma \,;\, \Psi \,;\, \Delta \vDash \Upsilon + \Delta', !A} \vDash'_{!}$$

We write $\Upsilon \nvDash \Delta$ if for no Υ' can we show $\Upsilon \vDash \Upsilon' + \Delta$. Armed with this matching judgement, we reconstruct the forward calculus of section 3.2 using affine contexts and other insights of section 3.3.1; In every rule of the logic requiring a particular form for the contexts in the premisses, we use our matching judgement in place of special contexts for the premisses. Additionally, the matching judgement obviates the left rules $\mathbf{1}\&L$, $\&\mathbf{1}L$ and $!L$, so we simply omit them.

Judgemental rules

$$\frac{}{\cdot \,;\, \cdot \,;\, [A]_0 \longrightarrow A} \text{init} \qquad \frac{\Gamma, A, A \,;\, \Psi \,;\, [\Delta]_w \longrightarrow \gamma}{\Gamma, A \,;\, \Psi \,;\, [\Delta]_w \longrightarrow \gamma} \text{factor}$$

$$\frac{[\Upsilon]_w \longrightarrow \gamma \quad [\Upsilon]_w \vDash (\Gamma \,;\, \Psi \,;\, \Delta) + A}{\Gamma, A \,;\, \Psi \,;\, [\Delta]_w \longrightarrow \gamma} \text{copy} \qquad \frac{[\Upsilon]_w \longrightarrow \gamma \quad \Upsilon \vDash (\Gamma \,;\, \Psi \,;\, \Delta) + A}{\Gamma \,;\, \Psi, A \,;\, [\Delta]_w \longrightarrow \gamma} \text{promote}$$

Note that in the “copy” and “promote” rules the principal formula A is considered as input. The “copy” rule, for instance, should be understood to mean that A may be copied into Γ if the required decomposition can be shown.

Multiplicative connectives

$$\frac{[\Upsilon]_w \longrightarrow \gamma \quad \Upsilon \vDash (\Gamma ; \Psi ; \Delta) + A, B}{\Gamma ; \Psi ; [\Delta, A \otimes B]_w \longrightarrow \gamma}\ \otimes L$$

$$\frac{[\Upsilon]_1 \longrightarrow \gamma \quad \Upsilon \vDash (\Gamma ; \Psi ; \Delta) + A}{\Gamma ; \Psi ; [\Delta, A \otimes B]_1 \longrightarrow \gamma}\ \otimes L_1 \qquad \frac{[\Upsilon]_1 \longrightarrow \gamma \quad \Upsilon \vDash (\Gamma ; \Psi ; \Delta) + B}{\Gamma ; \Psi ; [\Delta, A \otimes B]_1 \longrightarrow \gamma}\ \otimes L_2$$

$$\frac{\Gamma ; \Psi ; [\Delta]_{w_1} \longrightarrow A \quad \Gamma' ; \Psi' ; [\Delta']_{w_2} \longrightarrow B}{\Gamma, \Gamma' ; \Psi, \Psi' ; [\Delta, \Delta']_{w_1 \vee w_2} \longrightarrow A \otimes B}\ \otimes R \qquad \frac{}{\cdot ; \cdot ; [\cdot]_0 \longrightarrow \mathbf{1}}\ \mathbf{1}R$$

$$\frac{\Gamma ; \Psi ; [\Delta]_{w_1} \longrightarrow A \quad [\Upsilon]_{w_2} \longrightarrow \gamma \quad \Upsilon \vDash (\Gamma' ; \Psi' ; \Delta') + B}{\Gamma, \Gamma' ; \Psi, \Psi' ; [\Delta, \Delta', A \multimap B]_{w_1 \vee w_2} \longrightarrow \gamma}\ \multimap L$$

$$\frac{[\Upsilon]_w \longrightarrow \gamma \quad \Upsilon \vDash \Upsilon' + A \quad C \supseteq \gamma}{[\Upsilon']_w \longrightarrow A \multimap C}\ \multimap R \qquad \frac{[\Upsilon]_1 \longrightarrow C}{[\Upsilon]_1 \longrightarrow A \multimap C}\ \multimap R'$$

Additive Connectives

$$\frac{\Gamma ; \Psi ; [\Delta]_{w_1} \longrightarrow A \quad \Gamma' ; \Psi' ; [\Delta']_{w_2} \longrightarrow B}{\Gamma, \Gamma' ; \Psi \sqcup \Psi' ; [\Delta]_{w_1} + [\Delta']_{w_2} \longrightarrow A \,\&\, B}\ \&R \qquad \frac{}{\cdot ; \cdot ; [\cdot]_1 \longrightarrow \top}\ \top R$$

$$\frac{\begin{array}{l}[\Upsilon]_w \longrightarrow \gamma \\ \Upsilon \vDash (\Gamma ; \Psi ; \Delta) + A \quad B \neq \mathbf{1}\end{array}}{\Gamma ; \Psi ; [\Delta, A \,\&\, B]_w \longrightarrow \gamma}\ \&L_1 \qquad \frac{\begin{array}{l}[\Upsilon]_w \longrightarrow \gamma \\ \Upsilon \vDash (\Gamma ; \Psi ; \Delta) + B \quad A \neq \mathbf{1}\end{array}}{\Gamma ; \Psi ; [\Delta, A \,\&\, B]_w \longrightarrow \gamma}\ \&L_2$$

$$\frac{[\Upsilon]_w \longrightarrow A}{[\Upsilon]_w \longrightarrow A \oplus B}\ \oplus R_1 \qquad \frac{[\Upsilon]_w \longrightarrow B}{[\Upsilon]_w \longrightarrow A \oplus B}\ \oplus R_2$$

$$\frac{\begin{array}{l}[\Upsilon]_{w_1} \longrightarrow \gamma \\ \Upsilon \vDash (\Gamma ; \Psi ; \Delta) + A \quad \Gamma' ; \Psi' ; [\Delta']_{w_2} \longrightarrow \gamma'\end{array}}{\Gamma, \Gamma' ; \Psi \sqcup \Psi' ; [\Delta]_{w_1} + [\Delta']_{w_2}, A \oplus \mathbf{1} \longrightarrow \gamma \cup \gamma'}\ \oplus\mathbf{1}L$$

$$\frac{\begin{array}{r}[\Upsilon']_{w_2} \longrightarrow \gamma' \\ \Gamma ; \Psi ; [\Delta]_{w_1} \longrightarrow \gamma \quad \Upsilon \vDash (\Gamma' ; \Psi' ; \Delta') + B\end{array}}{\Gamma, \Gamma' ; \Psi \sqcup \Psi' ; [\Delta]_{w_1} + [\Delta']_{w_2}, \mathbf{1} \oplus B \longrightarrow \gamma \cup \gamma'}\ \mathbf{1}\oplus L$$

$$\frac{\begin{array}{ll}[\Upsilon]_{w_1} \longrightarrow \gamma & [\Upsilon']_{w_2} \longrightarrow \gamma' \\ \Upsilon \vDash (\Gamma ; \Psi ; \Delta) + A & \Upsilon' \vDash (\Gamma' ; \Psi' ; \Delta') + B\end{array}}{\Gamma, \Gamma' ; \Psi \sqcup \Psi' ; [\Delta]_{w_1} + [\Delta']_{w_2}, A \oplus B \longrightarrow \gamma \cup \gamma'}\ \oplus L \qquad \frac{}{\cdot ; \cdot ; [\mathbf{0}]_1 \longrightarrow \cdot}\ \mathbf{0}L$$

Exponential rules

$$\frac{[\Upsilon]_w \longrightarrow A \quad \Upsilon \vDash (\Gamma\,;\cdot\,;\cdot) + \cdot}{\Gamma\,;\cdot\,;[\cdot]_0 \longrightarrow\, !A}\ !R$$

Correctness. As expected, the comparatively complex nature of these rules makes soundness and completeness non-trivial properties. In fact, even simple statements of correspondence between the two calculi seem difficult to obtain. For a manageable description, we have to invoke the matching judgement.

$$\begin{array}{lll}
\Gamma\,;\Psi\,;[\Delta]_0 \longrightarrow C & \text{corresponds to} \quad \Gamma''\,;\Psi''\,;\Delta' \Longrightarrow C & \\
& \text{for any } \Gamma' \supseteq \Gamma,\ \Psi' \supseteq \Psi, \text{ and } \Delta' \supseteq \Delta & \\
& \text{such that } (\Gamma'\,;\Psi'\,;\Delta') \vDash (\Gamma''\,;\Psi''\,;\Delta) + (\Delta' \backslash \Delta) & (\text{lin}') \\
\Gamma\,;\Psi\,;[\Delta]_1 \longrightarrow \gamma & \text{corresponds to} \quad \Gamma'\,;\Psi'\,;\Delta' \Longrightarrow C & \\
& \text{for any } \Gamma' \supseteq \Gamma,\ \Psi' \supseteq \Psi,\ \Delta' \supseteq \Delta, \text{ and } C \supseteq \gamma & (\text{weak}')
\end{array}$$

To start with, we need to establish some properties of the matching judgement.

Lemma 3.18 (Bounding)**.** *If* $(\Gamma\,;\Psi\,;\Delta) \vDash (\Gamma'\,;\Psi'\,;\Delta') + \Delta''$*, then:*

1. $\Gamma' \subseteq \Gamma$, $\Psi' \subseteq \Psi$ *and* $\Delta' \subseteq \Delta$*; and*
2. $!(\Gamma\backslash\Gamma'), (\Psi\backslash\Psi')\ \&\ \mathbf{1}, (\Delta\backslash\Delta') \subseteq \Delta''$.

Proof. Induction on the structure of the derivation of $(\Gamma\,;\Psi\,;\Delta) \vDash (\Gamma'\,;\Psi'\,;\Delta') + \Delta''$. □

Additionally, we require a *matching lemma* that drives the completeness theorem.

Lemma 3.19 (Matching)**.** *If* $\Upsilon \Longrightarrow C$ *and* $\Upsilon \vDash (\Gamma\,;\Psi\,;\Delta) + \Delta'$ *then* $\Gamma\,;\Psi\,;\Delta,\Delta' \Longrightarrow C$*.*

Proof. Structural induction on the derivation of $\mathcal{M} :: \Upsilon \vDash (\Gamma\,;\Psi\,;\Delta) + \Delta'$. We illustrate with a pair of cases.

(i) The last rule of $\mathcal{M}$ is $\vDash_{\text{linear}}$, *i.e.*.

$$\frac{\Gamma\,;\Psi\,;\Delta \vDash (\Gamma'\,;\Psi'\,;\Delta') + \Delta''}{\Gamma\,;\Psi\,;\Delta, A \vDash (\Gamma'\,;\Psi'\,;\Delta') + \Delta'', A}$$

$\Gamma\,;\Psi\,;\Delta, A \Longrightarrow C$	hypothesis
$\Gamma\,;\Psi\,;\Delta \Longrightarrow A \multimap C$	$\multimap R$
$\Gamma'\,;\Psi'\,;\Delta', \Delta'' \Longrightarrow A \multimap C$	ind. hyp.
$\Gamma'\,;\cdot\,;A, A \multimap C \Longrightarrow C$	easily shown
$\Gamma'\,;\Psi'\,;\Delta', \Delta'', A \Longrightarrow C$	cut

(ii) The last rule of $\mathcal{M}$ is $\vDash_{\&\mathbf{1}}$, *i.e.*,

$$\frac{\Gamma\,;\Psi\,;\Delta \vDash (\Gamma'\,;\Psi'\,;\Delta') + \Delta''}{\Gamma\,;\Psi, A\,;\Delta \vDash (\Gamma'\,;\Psi'\,;\Delta') + \Delta'', A \,\&\, \mathbf{1}}$$

$\Gamma\,;\Psi, A\,;\Delta, A \Longrightarrow C$	hypothesis
$\Gamma\,;\Psi\,;\Delta, A \,\&\, \mathbf{1} \Longrightarrow C$	$\&\mathbf{1}L$
$\Gamma\,;\Psi\,;\Delta \Longrightarrow A \,\&\, \mathbf{1} \multimap C$	$\multimap R$
$\Gamma'\,;\Psi'\,;\Delta', \Delta'' \Longrightarrow A \,\&\, \mathbf{1} \multimap C$	ind. hyp.
$\Gamma'\,;\cdot\,;A \,\&\, \mathbf{1}, A \,\&\, \mathbf{1} \multimap C \Longrightarrow C$	easily shown
$\Gamma'\,;\Psi'\,;\Delta', \Delta'', A \,\&\, \mathbf{1} \Longrightarrow C$	cut

The other cases follow similarly. For the matching rules for the unrestricted context, we appeal to theorem 2.17 (extended with the affine zone). □

With these lemmas, we may now prove soundness and completeness of the forward calculus with respect to the backward calculus. Although the soundness theorem doesn't differ much from before, the completeness theorem has a somewhat unusual form, depending on the matching judgement. Nevertheless, we can prove these theorems purely by structural induction on the derivations. The difficulty in these theorems lies not in the inductions themselves, which follow straightforwardly, but rather in the choice of sufficiently strong induction hypotheses that make the inductions valid.

Theorem 3.20 (Soundness)**.**

1. *If* $[\Upsilon]_0 \longrightarrow C$ *then* $\Upsilon \Longrightarrow C$.
2. *If* $\Gamma\,;\Psi\,;[\Delta]_1 \longrightarrow \gamma$ *then* $\Gamma\,;\Psi\,;\Delta' \Longrightarrow C$ *for any* $\Delta' \supseteq \Delta$ *and* $C \supseteq \gamma$.

Proof. By structural induction on the derivation of $[\Upsilon]_w \longrightarrow \gamma$, similar to the proof of theorem 3.9, but using the matching and bounding lemmas as required. We omit the easy details. □

Theorem 3.21 (Completeness). *If the* $\mathbf{1}$*nf sequent* $\Gamma ; \Psi ; \Delta \Longrightarrow C$ *is derivable, then for some* $\Gamma' \subseteq \Gamma$, $\Psi' \subseteq \Psi$, *and* $\gamma \subseteq C$ *such that the following match holds*

$$(\Gamma' ; \Psi' ; \Delta) \vDash (\Gamma'' ; \Psi'' ; \Delta'') + \Delta'''$$

one of the following hold:

1. *either* $\Gamma'' ; \Psi'' ; [\Delta'', \Delta''']_0 \longrightarrow C$*;*
2. *or* $\Gamma'' ; \Psi'' ; [\Delta', \Delta''']_1 \longrightarrow \gamma$ *for some* $\Delta' \subseteq \Delta''$*.*

Proof. By structural induction on the derivation $\mathcal{D} :: \Gamma ; \Psi ; \Delta \Longrightarrow C$, using the bounding and matching lemmas. We have the following characteristic cases for the last rule in $\mathcal{D}$:

1. "init", "copy", "promote", $\mathbf{1}R$, $\top R$ or $!\,R$; these cases follow immediately because the rules in the forward and backward direction differ structurally only in the presence of the matching derivation, for which we invoke the matching lemma.
2. $\mathbf{1}\&L$ or $\&\mathbf{1}L$; for example

   $$\frac{\mathcal{D}' :: \Gamma ; \Psi, A ; \Delta \Longrightarrow C}{\mathcal{D} :: \Gamma ; \Psi ; \Delta, A \,\&\, \mathbf{1} \Longrightarrow C}\ \mathbf{1}\&L$$

   Invoking the bounding lemma (case 1), assume given $\Gamma_1 \cup \Gamma_2 \subseteq \Gamma$ and $\Psi_1, \Psi_2 \subseteq (\Psi, A)$. Then, we have
   (a) if $\Gamma_1 ; \Psi_1 ; [\Delta, !\,\Gamma_2, \Psi_2 \,\&\, \mathbf{1}]_0 \longrightarrow C$, then
      i. if $\mathbf{1} \,\&\, A \in \Psi_2 \,\&\, \mathbf{1}$, then we satisfy case (1);
      ii. otherwise, $\Psi_1, \Psi_2 \subseteq \Psi$ and we satisfy case (1).
   (b) otherwise, $\Gamma_1 ; \Psi_1 ; [\Delta', !\,\Gamma_2, \Psi_2 \,\&\, \mathbf{1}]_1 \longrightarrow C$ for some $\Delta' \subseteq \Delta$; the above argument still applies, except now we satisfy case (2) instead of (1).
3. Other rules require a similar but simpler enumeration of possibilities. □

Lest the completeness theorem give the impression that matching as a judgement makes forward reasoning unusably complex, we can restate the correspondences to the backward calculus in simpler terms using the bounding lemma.

$$\Gamma ; \Psi ; [\Delta]_0 \longrightarrow C \qquad \text{corresponds to} \qquad \Gamma' ; \Psi' ; \Delta' \Longrightarrow C$$

for any $\Gamma' \supseteq \Gamma$, $\Psi' \supseteq \Psi$, and for $\Delta' \supseteq \Delta$ where every element of $\Delta' \backslash \Delta$ has one of the forms $A \,\&\, \mathbf{1}$, $\mathbf{1} \,\&\, A$, or $!\,A$; and

$$\Gamma ; \Psi ; [\Delta]_1 \longrightarrow \gamma \qquad \text{corresponds to} \qquad \Gamma' ; \Psi' ; \Delta' \Longrightarrow C$$

for any $\Gamma' \supseteq \Gamma$, $\Psi' \supseteq \Psi$, $\Delta' \supseteq \Delta$, and $C \supseteq \gamma$.

3.3.3 Proof extraction

The previous section gives us a means of constructing proofs of propositions in **1**nf, but that still leaves the question of proving the original sequent. As mentioned already, we first construct the **1**nf of the given input sequent. Therefore, the problem is to construct a witness for the original sequent given a proof of the **1**nf sequent. A first attempt might be to convert the sequent derivation for the **1**nf sequent into that of the original sequent as follows: start with the proof of the **1**nf sequent, and replay the rewrites used to convert the sequent to **1**nf in reverse, making local modifications to the derivations as needed, to recover the derivation for the original goal sequent. For example, suppose we have a derivation $\mathcal{D} :: \Gamma \,;\Psi\,;\Delta \Longrightarrow A$. We can then easily see that $\otimes R(\mathcal{D}, \mathbf{1}R\,; A \otimes \mathbf{1}) :: \Gamma\,;\Psi\,;\Delta \Longrightarrow A \otimes \mathbf{1}$.

This attempt breaks down on the left. Given a derivation $\mathcal{D} :: \Gamma, A\,;\Psi\,;\Delta \Longrightarrow C$, there is no way to *locally* convert it to a derivation of $\mathcal{D} :: \Gamma, A \otimes \mathbf{1}\,;\Psi\,;\Delta \Longrightarrow C$. The only way to proceed is to cut out the A using a derivation of $\Gamma, A \otimes \mathbf{1}\,;\cdot\,;\cdot \Longrightarrow A$ (easy to achieve), but, being a cut, will cause a global change to the given derivation $\mathcal{D}$. Although straightforward in theory, we decided not to implement full cut-elimination for sequent derivations because we never explicitly construct the derivation $\mathcal{D}$ in full. The syntax for $\mathcal{D}$ uses a labelled representation, which would require renaming every time a sequent is considered as a premiss of an inference rule. The information we do maintain is insufficient for cut-elimination.

Instead, we implement the proof extraction at the level of natural deduction proof terms. Recall that our overall goal is to present a natural deduction proof to the user. Suppose we know $(\Gamma)_\mathbf{1}\,;(\Delta)_\mathbf{1} \vdash M : (C)_\mathbf{1}$. The question then is how to construct N from M such that $\Gamma\,;\Delta \vdash N : C$. Since the **1** normal form transformation in definition 3.12 works solely with equivalences, we know that for every proposition A, we can define two functions f_A and g_A such that

$$\Gamma\,;\cdot \vdash f_A : A \multimap (A)_\mathbf{1} \qquad \text{and} \qquad \Gamma\,;\cdot \vdash g_A : (A)_\mathbf{1} \multimap A.$$

Therefore, given

$$u_1 : (A_1)_\mathbf{1}, \ldots\,; v_1 : (B_1)_\mathbf{1}, \ldots \vdash M : (C)_\mathbf{1}$$

We have:

$$u_1' : A, \ldots ; v_1' : B_1, \ldots \vdash g_C \, [f_{A_1} \, u_1'/u_1, \ldots, f_{B_1} \, v_1'/v_1, \ldots]M : C.$$

Of course, this resulting proof term will not be normal, but it can be normalised if needed.

3.4 Optimization: irredundancy

The optimization in the previous section is motivated by observing the behaviour of the **1** connective and an attempt to control the weakening induced by it. In order to achieve this, we had to implicitly generalize the notion of subsumption of sequents to allow for such locally affine resources. We already have a notion of subsumption of sequents: definition 3.7. If forward reasoning is to produce new knowledge, one obvious restriction to ensure that the rules of our calculus do not produce sequents that will be immediately subsumed by one of the premisses of the rule. We call this an *irredundancy criterion* for inference rules.

Definition 3.22 (Irredundancy criterion)**.** *A rule*

$$\frac{\Gamma_1 \,;\, [\Delta_1]_{w_1} \longrightarrow \gamma_1 \quad \cdots \quad \Gamma_n \,;\, [\Delta_n]_{w_n} \longrightarrow \gamma_n}{\Gamma_0 \,;\, [\Delta_0]_{w_0} \longrightarrow \gamma_0}$$

is said to be irredundant *if for no $i \in 1 \ldots n$ is* $(\Gamma_i \,;\, [\Delta_i]_{w_i} \longrightarrow \gamma_i) \prec (\Gamma_0 \,;\, [\Delta_0]_{w_0} \longrightarrow \gamma_0)$.

It turns out that not all rules of the calculus of section 3.2.2 are irredundant. A simple example is $!\,L$, wherein we are allowed to conclude (assuming $A \notin \Gamma$):

$$\frac{\Gamma \,;\, [\Delta]_1 \longrightarrow \gamma}{\Gamma \,;\, [\Delta, !\,A]_1 \longrightarrow \gamma} \; !\,L$$

In this case, the conclusion of the rule is immediately subsumed by the strictly stronger premiss, and therefore provides no new information. For proof search, it is important to eliminate such fruitless applications of inference rules.

Redundancy also shows up in a subtler form when rules are chained together. For example, the following are two ways to apply $\otimes L$ to the sequent $\Gamma \,;\, [\Delta, A, B]_1 \longrightarrow \gamma$ using the resources A and B:

$$\frac{\Gamma \,;\, [\Delta, A, B]_1 \longrightarrow \gamma}{\Gamma \,;\, [\Delta, A \otimes B]_1 \longrightarrow \gamma} \qquad \text{and} \qquad \frac{\Gamma \,;\, [\Delta, A, B]_1 \longrightarrow \gamma}{\Gamma \,;\, [\Delta, A \otimes B, B]_1 \longrightarrow \gamma}$$

In the first case both resources are consumed, but in the second case only the resource A is consumed. It is clear that the conclusion of the first rule subsumes the conclusion of the second rule; thus, it is only necessary to allow $\otimes L$ applications of the first kind.

To prevent applications of the second kind, we have to introduce a new kind of precondition on inference rules: a *negative-existence condition*. That is, we modify the $\otimes L'$ rule from before into:

$$\frac{\Gamma\,;[\Delta, A]_1 \longrightarrow \gamma \quad B \notin \Delta}{\Gamma\,;[\Delta, A \otimes B]_1 \longrightarrow \gamma}\,\otimes L_1' \qquad \text{and} \qquad \frac{\Gamma\,;[\Delta, B]_1 \longrightarrow \gamma \quad A \notin \Delta}{\Gamma\,;[\Delta, A \otimes B]_1 \longrightarrow \gamma}\,\otimes L_2'$$

In other words, we *require* one of the operands to be present and the other to be absent if we are to use the implicit weakenability of weak sequents. If both operands are present, then we consume *both* of them in the standard $\otimes L$ rule. In the rest of this section, we will perform a similar optimization on every rule of the calculus. This will not only create a calculus where all rules are irredundant, but also provide precise matching conditions to minimize redundant creation of sequents.

As already mentioned, we refine the $\otimes L$ rules to prevent creation of redundant conclusions.

$$\frac{\Gamma\,;[\Delta, A, B]_w \longrightarrow \gamma}{\Gamma\,;[\Delta, A \otimes B]_w \longrightarrow \gamma}\,\otimes L \qquad \frac{\Gamma\,;[\Delta, A]_1 \longrightarrow \gamma \quad B \notin \Delta}{\Gamma\,;[\Delta, A \otimes B]_1 \longrightarrow \gamma}\,\otimes L_1' \qquad \frac{\Gamma\,;[\Delta, B]_1 \longrightarrow \gamma \quad A \notin \Delta}{\Gamma\,;[\Delta, A \otimes B]_1 \longrightarrow \gamma}\,\otimes L_2'$$

Similarly for $\mathbf{1}L$, where we no longer even need to consider the case of weak sequents as the conclusion would be immediately subsumed.

$$\frac{\Gamma\,;[\Delta]_0 \longrightarrow C}{\Gamma\,;[\Delta, \mathbf{1}]_0 \longrightarrow C}\,\mathbf{1}L$$

The right rules $\otimes R$ and $\mathbf{1}R$ require no modifications because in each case the proposition on the right of the sequent arrow changes, ensuring that the conclusion can not be subsumed by a premiss according to definition 3.7.

For $\multimap R$ for weak sequents, we have the following possibilities: if the antecedent is present as a resource in the premiss, we can be lax about the conclusion:

$$\frac{\Gamma\,;[\Delta, A]_1 \longrightarrow \gamma \quad \gamma \subseteq B}{\Gamma\,;[\Delta]_1 \longrightarrow A \multimap B}\,\multimap R_1$$

If the antecedent is absent, then the right hand side must be present to prevent a redundant conclusion:

$$\frac{\Gamma\,;[\Delta]_1 \longrightarrow B \quad A \notin \Delta}{\Gamma\,;[\Delta]_1 \longrightarrow A \multimap B}\,\multimap R_2$$

Note here the side condition $A \notin \Delta$; if this were not required, then we would be able to apply $\multimap R$ in two different ways, giving a stronger conclusion in one case as follows:

$$\frac{\Gamma\,;[\Delta, A]_1 \longrightarrow B}{\Gamma\,;[\Delta]_1 \longrightarrow A \multimap B} \qquad \text{and} \qquad \frac{\Gamma\,;[\Delta, A]_1 \longrightarrow B}{\Gamma\,;[\Delta, A]_1 \longrightarrow A \multimap B}$$

For $\multimap L$ on the principal resource $A \multimap B$, there are two important cases to consider. In the first case, the premiss with resource B is strong, i.e.,

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow A \quad \Gamma'\,;[\Delta', B]_0 \longrightarrow C}{\Gamma, \Gamma'\,;[\Delta, \Delta', A \multimap B]_w \longrightarrow C}$$

Note that the right hand side of the first premiss is forced to be present, for if it were absent (which would require $w = 1$), then the conclusion is weaker than the premiss and the rule is redundant. The other case is where the premiss with resource B is weak, i.e., we have:

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow A \quad \Gamma'\,;[\Delta', B]_1 \longrightarrow \gamma}{\Gamma, \Gamma'\,;[\Delta, \Delta', A \multimap B]_w \longrightarrow \gamma}$$

(Clearly B must be present in the premiss, or the conclusion would be subsumed by this premiss.) Now we obtain an odd interaction with the first premiss if $\Delta = \Delta'', B$:

$$\frac{\Gamma\,;[\Delta'', B]_w \longrightarrow A \quad \Gamma'\,;[\Delta', B]_1 \longrightarrow \gamma}{\Gamma, \Gamma'\,;[\Delta'', B, \Delta', A \multimap B]_w \longrightarrow \gamma}$$

The conclusion is now subsumed by the second premiss! We must therefore ensure that B does not occur as a resource in the *first* premiss. Thus we obtain a somewhat strange form of the irredundant $\multimap L$ rule:

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow A \quad \Gamma'\,;[\Delta', B]_{w'} \longrightarrow \gamma \quad (w' = 0 \vee B \notin \Delta)}{\Gamma, \Gamma'\,;[\Delta, \Delta', A \multimap B]_{w \vee w'} \longrightarrow \gamma}\multimap L$$

The additive rules are already irredundant. For the exponential rules, specifically for $!L$, we have to ensure that if we implicitly weaken the unrestricted context then the conclusion of the rule is not subsumed by the premiss. This is the case only if the conclusion is not weak. Thus, we obtain:

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow \gamma \quad (A \in \Gamma \vee w = 0)}{\Gamma \backslash A\,;[\Delta, !A]_w \longrightarrow \gamma}\,!L$$

Theorem 3.23 (Irredundant formulation)**.** *The modified rules presented in this section are all irredundant.*

Proof. Simple inspection. □

Completeness of the irredundant formulation is rather an obvious property, because there are no cases in the proof of theorem 3.11 that require the use of a redundant rule.

Theorem 3.24 (Completeness of irredundant forward derivations)**.**
Suppose $\Gamma \mathrel{;} \Delta \Longrightarrow C$*. Then,*

(a) either $\Gamma' \mathrel{;} [\Delta]_0 \longrightarrow C$*,*
(b) or $\Gamma' \mathrel{;} [\Delta']_1 \longrightarrow \gamma$

for some $\Gamma' \subseteq \Gamma$*,* $\Delta' \subseteq \Delta$ *and* $\gamma \subseteq C$*.*

Proof. Same proof as for theorem 3.11. □

Irredundancy manifests in an implementation in the form of strong negative existence checks on the applicability of a given rule to a sequent. In the propositional case it is relatively straightforward to ensure that a given proposition does not occur in the input sequent. When extended to the first-order calculus, however, these negative existence conditions are not as straightforward as they require the use of unification. As unification is an expensive operation to perform too often, it turns out that it is simply better to pay the penalty of (knowingly) creating redundant sequents sometimes. Furthermore, in the presence of free variables in the calculus, factoring is no longer a unitary operation, i.e., there are many incomparable factors of a given sequent. It is impossible to ensure that factoring produces irredundant sequents without an exhaustive test of all factors of a sequent for redundancy, which entirely misses the point of these optimizations intended as tight matching conditions on rule applicability. Lastly, when we allow derived rule creation using focused derivations in chapter 6, we find that the negative existence conditions considerably complicate the presentation of the focusing calculus and the proof of its completeness. Unsurprisingly, they are very tricky to implement for multi-premiss derived rules; as seen for $\multimap L$, the non-existence condition requires interactions between entirely disjoint branches of a derivation, and the situation is already enormously complex with just two interacting $\multimap L$ rules. Therefore, for both theoretical and practical reasons, we shall not use irredundant calculi further in this work. Note that we always check if a newly generated sequent is subsumed by an earlier sequent before keeping the sequent for future rule applications, so irredundancy is always ensured by our search procedure.

3.5 Historical review

Resource management in backward reasoning has a relatively long history given the age of linear and sub-structural logics, with the earliest identification of this issue in proof search dating back to the work of Harland and Pym in 1991 [48]. Subsequently, in the settings of backward proof search and logic programming in the (linear) uniform fragment, Harland and Winikoff [56], Cervesato, Hodas and Pfenning [23], and most recently Harland and Pym [49] have provided solutions for the resource management problem. The weakening annotation introduced in this paper bears a strong resemblance to a similar notation in [23], although the interpretation differs considerably because of the different nature of forward search.

To the best of our knowledge, resource management in the forward direction has not received any satisfactory treatment in the literature (aside from the present work). The oldest work on forward-reasoning in linear logic, due to Mints [83], discusses a kind of resolution calculus for linear logic; however, his resolution calculus differs significantly from the usual notion of resolution because of the inclusion of *axiomatic clauses* of the form $\Gamma, \top$. The context Γ in these clauses remains unspecified, so for an implementation it becomes necessary to restrict the use of such axioms, specifically by discovering permutations in the resolutions steps that allow pushing these axioms downwards. The resolution calculus of Mints suffers from an additional problem, arising from the inclusion of explicit weakening rules for the classical "why not" modality, ?. Tammet [110] has performed a fuller examination of the allowable permutations, together with more efficient treatment of weakening and exponentials, but both Mints and Tammet describe what we now understand as resource management problems in terms of search strategies. In other words, their calculi lack the explicit examination of resource management issues in the proof theory, in the style of Cervesato *et al* [23] or Harland and Pym [49]. We have not encountered any other investigations of forward reasoning in linear logic in our literature survey.

Chapter summary *In this chapter we have presented the first forward sequent calculus for the propositional fragment of the logic of chapter 2. We have further presented two possible optimisations: one detects certain idiomatic uses of* **1** *and handles them using a specially moded context, and the other makes peephole optimisations to the inference rules to prevent creating redundant conclusions. The development of these calculi are motivated by resource-management considerations.*

The next chapter will present the inverse method that implements the forward calculus of this section.

Chapter 4

The Inverse Method

The forward calculi in the previous chapter are designed to be used in a forward search procedure, but there are several details of the procedure that are not illuminated by the calculus itself. In this chapter, we shall lay out the details of the inverse method search procedure that uses the calculus of the previous section. We continue in the propositional fragment of the previous chapter. When we extend the forward calculus with quantifiers (in chapter 5), we shall describe the modifications required to the algorithm outlined in this chapter then.

4.1 The Subformula Property

The key technical property that makes the inverse method possible is the *subformula property*. Stated simply, in cut-free sequent calculus proofs, we need to consider only sequents composed of subformulas of the goal sequent. To illustrate, assuming we have sequents containing A and B, then we never consider a rule to infer a sequent about $A \mathbin{\&} B$ from these sequents, unless $A \mathbin{\&} B$ occurs as a subformula of the goal sequent. Formally, we present this property property in terms of a *subformula relation* for propositions. To describe the subformula relation in its strongest form, we decorate subformulas with certain marks:

1. *Sign* (also known as *polarity*), which we write as a superscript $^+$ ("positive") or $^-$ ("negative"), written schematically as $^\pm$. The operands of all binary connectives

inherit the sign of the formula, with the exception of $A \multimap B$, for which A receives the opposite sign. Formulas to the right of the sequent arrow receive the positive sign, and those on the left the negative sign. Thus, these signs indicate the side of the sequent arrow where the formula occurs as a principal formula when it is inferred.

2. *Availability*, which we write as a subscript $_!$ ("unrestricted") or $_\cdot$ ("linear"), and schematically as $_\mathrm{a}$. Top-level formulas in the unrestricted context, and operands of ! receive this decoration, but the subformulas do not inherit the decoration. These signs, therefore, determine whether the formula is allowed to occur in the unrestricted context, and thus serve as a guide for the copy rule.

Definition 4.1 (Decorated propositions)**.** *A decorated proposition is of the form $A_a^\pm$ where a is either $!$ or $\cdot$ and $\pm$ is either $+$ or $-$. A partially decorated proposition may omit either the sign or the availability decoration.*

Definition 4.2. *A (partially) decorated context consists of (partially) decorated propositions of the same sign and availability. We appropriate the notations $^\pm$ and $_a$ for (partially) decorated contexts.*

Definition 4.3. *A decorated sequent is of the form: $\Gamma_!^- \,;\, \Delta_\cdot^- \Longrightarrow C_\cdot^+$.*

Definition 4.4 (Decorated subformula relation)**.** *The decorated subformula relation $\leq$ between decorated propositions is the reflexive-transitive closure of the following cases.*

$$
\begin{array}{lll}
A_\cdot^\pm \leq (A * B)_a^\pm & B_\cdot^\pm \leq (A * B)_a^\pm & \ldots * \in \{\otimes, \&, \oplus\} \\
A_\cdot^\mp \leq (A \multimap B)_a^\pm & B_\cdot^\pm \leq (A \multimap B)_a^\pm & \\
A_!^\pm \leq (!A)_a^\pm & A_\cdot^\pm \leq A_!^\pm &
\end{array}
$$

(We abuse notation slightly by writing $A^\pm \leq B^\pm$ for the parallel pair $A^+ \leq B^+$ and $A^- \leq B^-$ together, and use $\mp$ as the "opposite" of $\pm$ for the antiparallel case.) On the atoms and propositional constants it is the identity. We assume the standard forgetful restrictions of this relation to partially decorated propositions, and the pointwise extension of this relation to sets and collections of (partially) decorated propositions.

Note that the last case of the definition of $\leq$ makes the linear decoration subordinate to the unrestricted decoration. This is informed by judgemental considerations: an unrestricted resource may be copied arbitrarily often into a linear resource.

Definition 4.5 (Decorated subsequent relation)**.** *A decorated sequent* $s_1 = \Gamma^-_! \,;\, \Delta^-_. \Longrightarrow C^+_.$ *is a subsequent of the sequent* $s_2 = \Gamma'^-_! \,;\, \Delta'^-_. \Longrightarrow C'^+_.$*, written* $s_1 \leq s_2$ *if:*

$$\Gamma'^-_! \cup \Delta'^-_. \cup C'^+_. \leq \Gamma^-_! \cup \Delta^-_. \cup C^+_.$$

Using the subformula relation, we may state the subformula property of the sequent calculus in the strongest form as follows:

Theorem 4.6 (Subformula property)**.** *If* $\Gamma' \,;\, \Delta' \Longrightarrow C'$ *appears in a proof of* $\Gamma \,;\, \Delta \Longrightarrow C$*, then:*

$$\Gamma'^-_! \cup \Delta'^-_. \cup C'^+_. \leq \Gamma^-_! \cup \Delta^-_. \cup C^+_.$$

Proof. Induction on the structure of $\mathcal{D} :: \Gamma \,;\, \Delta \Longrightarrow C$ by splitting cases on the final rule used. In each case we assume that the condition holds for the conclusion of the rule, and then show that it holds for the premisses. For the signs, the argument is fairly straightforward. For availability, we have to sometimes appeal to the fact that an unrestricted decoration may be relaxed to a linear decoration. □

By theorem 3.11, sequents in the forward calculus contain a subset of formulas in the backward calculus. Thus,

Corollary 4.7. *If* $\Gamma' \,;\, [\Delta']_w \longrightarrow \gamma'$ *appears in a proof of* $\Gamma \,;\, [\Delta]_{w'} \longrightarrow \gamma$*, then:*

$$\Gamma'^-_! \cup \Delta'^-_. \cup \gamma'^+_. \leq \Gamma^-_! \cup \Delta^-_. \cup \gamma^+_.$$

Proof. Directly from theorem 4.6. □

4.1.1 Labelling and specialized rules.

The subformula property gives us the core of the inverse method procedure. We start with initial sequents of the form $\cdot \,;\, \cdot\, [p]_0 \Longrightarrow p$, where the atom p occurs as both a positive and a negative subformula of the decorated goal sequent. Since we need some way to refer to subformulas of the goal sequent, we label label all subformulas with new fresh (propositional) labels, which we write using the propositional atomic variables u, l, r, etc.

Definition 4.8 (Notational definition)**.** *We write* $l \# A$ *to denote that* l *is the label for the proposition* A*.*

Definition 4.9 (Labelling). *Given a goal sequent $A_1, \ldots, A_m ; B_1, \ldots, B_n \Longrightarrow C$, we define the following top-level labels: $u_i \# A_i$, $l_j \# B_j$ and $r \# C$. Furthermore, we label every non-atomic subformula of the sequent using a unique label.*

We also specialize all rules to these labels by means of a pre-processing stage before entering into the main search procedure. We do not maintain general rules for conjunction, disjunction *etc.*, but instead have a version of every rule for every label, with the label taking the role of the principal formula.

4.1.2 Sequent representation

Represented sequents consist of a collection of labels of resources and the label of the goal. More precisely, these collections are multisets. For each label, we maintain a *multiplicity* of that label, standing for the number of instances of that particular subformula in the sequent. These multiplicities are per subformula of the goal sequent, not per proposition, as identical non-atomic subformulas that originate from different parts of the sequent will receive different labels. However, this artificial restriction is removed for the atomic propositions, for which the multiplicity reflects the number of instances of that proposition in the sequent. This is necessary for initial sequents, as the same label must occur on both sides of the sequent arrow.

The implementation of contexts of labels is guided by several considerations. The most common operation in the theorem prover is querying for the existence of a given resource or label in the sequent. Therefore, these context representations must support efficient lookup and update of the multiplicities of resources. Another common operation in the forward direction is the join of contexts of resources multiplicatively. Thus, the collections must support efficient merge operations also. However, in the linear theorem prover we are entirely uninterested in the *ordering* of the resources and labels in the contexts (though, for presentational and debugging purposes, it is good to be able to consistently linearise a context in a predictable order).

The implementation of contexts we choose is a binary map between labels and their multiplicities. The underlying map data structure is a dynamically reorganizing persistent splay tree [107] with the following relaxation: merges are not required to guarantee amortized constant access to the elements of the tree. Labels are selected by generating uniformly hashing the string representation of their names, and they are ordered by their hash values.

In addition to the multiplicity of labels, we also maintain the current weak flags for the linear resources and the judgement (*goal* or *poss*) on the right of the sequent arrow. We do *not*, however, maintain any hypothesis variables in sequents.

Definition 4.10 (Sequent representation)**.** *A forward sequent is represented as follows*

$$\underbrace{u_1 \# A_1, \ldots, u_m \# A_n}_{\Gamma} \quad ; \quad [\underbrace{l_1^{k_1} \# B_1, \ldots, l_n^{k_n} \# B_k}_{\Delta}]_w \quad \longrightarrow \quad \underbrace{\begin{cases} r \# C \\ \cdot \end{cases}}_{\gamma}$$

where each u_i, l_j and r are labels of subformulas of the final goal sequent, k_i are the multiplicities of the corresponding labels, and w is the weakening flag on the linear context Δ. Define mult(Γ, l) as the multiplicity of label l in context Γ, and similarly for Δ.

For the rest of this work, the propositional parts (i.e., of the form $\# A$) will be omitted when not particularly relevant.

4.1.3 Subsumption

In the saturation-based search that we use in the forward direction, there is a so-called *conjunctive non-determinism* in selecting sequents for applying rules. It is therefore critically important that the database of sequents available as candidates for rule application contains as little redundancy as possible. We therefore have to resort to checking for sequent subsumption whenever a new sequent is created; this is sometimes referred to as *forward subsumption*.

The complications in the linear setting lie in handling the linear context for weak sequents, for which we allow subsumption of sequents with weaker contexts even though we don't have an admissible structural theorem for weakening the linear context of weak sequents. Nevertheless, we don't lose completeness because we can always use the stronger sequent for any purpose the weaker sequent might serve.

In implementing subsumption, it is much more important to detect failures as early as possible because the vast majority of sequent comparisons will not yield a positive subsumption match. The usual strategy employed is to perform a sequence of *hierarchical tests* that imply subsumption if and if they all succeed. For the propositional case these tests are, for the most part, easily done, but because we will want to extend them to the first-order case in the next chapter, it is still instructive to see the order.

Definition 4.11 (Hierarchical subsumption tests). *A sequent* $s_1 = \Gamma\,;[\Delta]_w \longrightarrow \gamma$ *does* not *subsume* $s_2 = \Gamma'\,;[\Delta']_{w'} \longrightarrow \gamma'$, *written* $s \ntrianglelefteq s_2$ *if*

1. $w = 0$ *and* $w' = 1$; *or*
2. $\gamma \nsubseteq \gamma'$; *or*
3. $\Delta \nsubseteq \Delta'$; *or*
4. $\Gamma \nsubseteq \Gamma'$; *or*
5. $s_1 \nprec s_2$.

Here $\Gamma \subseteq \Gamma'$ *if for every* $l \in \mathrm{dom}(\Gamma)$, $\mathit{mult}(\Gamma, l) \leq \mathit{mult}(\Gamma', l)$, *and similarly for* Δ. *As usual,* $\gamma \subseteq \gamma'$ *if and only if* $\gamma = \cdot$ *or* $\gamma = \gamma' = r \mathbin{\#} C$.

Since we are only interested in detecting failures, we never use the positive version ($\trianglelefteq$) of this test. Operationally, the cases of the test are treated as a large `if-then-else` statement. We also have the following trivial theorem whose proof we omit:

Theorem 4.12 (Completeness of hierarchical tests). *For any pair of sequents* s_1 *and* s_2, $s_1 \prec s_2$ *(i.e.,* s_1 *subsumes* s_2*) if and only if not* $s_1 \ntrianglelefteq s_2$. □

4.1.4 Factoring

The rule "factor" has to be explicitly implemented in the forward direction because the input unrestricted (and weak linear) contexts are not guaranteed to be equal in additive rules. The important question to answer in an implementation is when to apply the factoring rules. Factoring too often may put a large strain on the main loop of the proving engine, wasting time tidying sequents that may never yield a proof. Contrarily, keeping sequents unfactored can lead to increasing the amount of non-determinism in rule application (as unfactored labels can be matched in a rule application several times). We have experimented with both strategies and found that delaying factoring rarely has a performance benefit in the absence of focusing, and delayed factoring is very complex to implement in the presence of first-order quantifiers. In the propositional case, factoring is actually extremely easy to implement.

Definition 4.13 (Factoring)**.** *Given two contexts* Γ_1 *and* Γ_2*, their common factor, written* $\Gamma_1 \sqcup \Gamma_2$*, is defined to be the context with domain* $\mathrm{dom}(\Gamma_1) \cup \mathrm{dom}(\Gamma_2)$ *such that for every* $l \in \mathrm{dom}(\Gamma)$*,* $\mathit{mult}(\Gamma, l) = \mathit{max}(\mathit{mult}(\Gamma_1, l), \mathit{mult}(\Gamma_2, l))$*.*

Note that this is the same notion as the least upper bound of multisets as defined in sec. 3.2.2.

In our implementation we do not have an arbitrary rule "factor", but rather eagerly factor contexts whenever needed. Thus, for example, the $\otimes R$ rule is implemented as

$$\frac{\Gamma_1 \,;\, [\Delta_1]_{w_1} \longrightarrow r_A \,\#\, A \;\mathit{goal} \quad \Gamma_2 \,;\, [\Delta_2]_{w_2} \longrightarrow r_B \,\#\, B \;\mathit{goal}}{\Gamma_1 \sqcup \Gamma_2 \,;\, [\Delta_1, \Delta_2]_{w_1 \vee w_2} \longrightarrow r \,\#\, A \otimes B \;\mathit{goal}}$$

As a side-effect, the multiplicity of every label in the unrestricted context is always at-most 1, i.e., the unrestricted context is interpreted as a set instead of a multiset. An implementation can make use of this property to select a specialised set data structure for its representation, but in our implementation we have chosen to treat all contexts uniformly. The following proof of completeness can be formalised if necessary, but it is such an obvious property that we have not taken this additional step.

Fact 4.14 (Completeness of eager factoring)**.** *The version of the forward calculus with eager factoring is complete with respect to the calculus with explicit "factor" rules.* □

4.1.5 Rules and rule application

As mentioned earlier, rules are precomputed to the specific labels of the subformulas of the goal sequent. If the positive proposition $r \,\#\, A \otimes B$ occurs in the goal sequent, and $r_A \,\#\, A$ and $r_B \,\#\, B$ are the labels of the operands, then the following rule will be generated for this label:

$$\frac{\Gamma_1 \,;\, [\Delta_1]_{w_1} \longrightarrow r_A \quad \Gamma_2 \,;\, [\Delta_2]_{w_2} \longrightarrow r_B}{\Gamma_1, \Gamma_2 \,;\, [\Delta_1, \Delta_2]_{w_1 \vee w_2} \longrightarrow r} \otimes R_r \tag{4.1}$$

To apply a rule to a given input sequent, we have to first *match* a premiss of the rule to the input sequent. Matching is non-deterministic: the same sequent can match the same premiss in a number of different ways. To formalise this, we need a definition of a sequent *schema*

Definition 4.15 (Sequent schema)**.** *A sequent schema is of the form:*

$$\underbrace{u_1, \ldots, u_m}_{\Gamma} \quad ; \quad \underbrace{[l_1^{k_1}, \ldots, l_n^{k_n}]_\omega}_{\Delta} \quad \longrightarrow \quad \underbrace{\begin{cases} r \\ \cdot \end{cases}}_{\gamma}$$

where ω is 0, 1 or $\cdot$, and the remaining components have the same meanings as in definition 4.10.

Definition 4.16 (Matching)**.** *We say that a schema $\sigma = \Gamma_s\,;[\Delta_s]_{\omega_s} \longrightarrow \gamma_s$* matches *an input sequent $s = \Gamma\,;[\Delta]_w \longrightarrow \gamma$ if*

1. $\Gamma_s \subseteq \Gamma$,
2. $\Delta_s \subseteq \Delta$,
3. $\omega_s \subseteq w$ *(i.e., $\omega_s = \cdot$ or $\omega_s = w$), and*
4. $\gamma_s \subseteq \gamma$.

A result of the match, written $\sigma \mid s$, is a sequent $\Gamma_r\,;[\Delta_r]_{w_r} \longrightarrow \gamma_r$ for which

1. $\Gamma_r = \Gamma \backslash \Gamma_s$
2. $\Delta_r = \Delta \backslash \Delta_s$
3. $w_r = w$ *if* $\omega_s = \cdot$ *and* $w_r = \omega_s$ *otherwise*
4. $\gamma_r = \gamma$

Results may not be unique, in which case $\sigma \mid s$ refers non-deterministically to any result.

A specialised rule such as $\otimes R_r$ above (4.1) is treated as having two components. The top half is where a pair of schemas is used by a *matching engine* against input sequents to see if the rule is applicable, and a bottom half that uses the results of the match to actually compute the conclusion sequent. We write the rule (4.1) therefore as:

$$\frac{\begin{array}{cc} & \cdot\,;[\cdot]_\cdot \longrightarrow r_A\ true \mid s_1 = \Gamma_1\,;[\Delta_1]_{w_1} \longrightarrow \cdot\ goal \\ s_1 \quad s_2 & \cdot\,;[\cdot]_\cdot \longrightarrow r_B\ true \mid s_2 = \Gamma_2\,;[\Delta_2]_{w_2} \longrightarrow \cdot\ goal \end{array}}{\Gamma_1 \sqcup \Gamma_2\,;[\Delta_1, \Delta_2]_{w_1 \vee w_2} \longrightarrow r\ goal}\ \otimes R_r$$

In other words, the schema exposes the components of the input sequent that *must be* present for the rule to be applicable.

4.1.6 Search procedure

Finally, a brief summary of the search procedure:

1. Label all subformulas of the goal sequent, and decorate using signs and availabilities.
2. Determine all initial sequents for atomic formulas with both signs.
3. Specialize all left rules for negative subformulas, all right rules for positive subformulas, and instances of the "copy" rule for unrestricted subformulas.
4. Starting from the initial sequents, apply the inference rules in any order that is guaranteed to saturate the search space. Add new facts to a database used for forward subsumption. As an optimization, after applying all possible rules for a given sequent, mark the sequent as "old", and never consider it for generating new facts again. Thus, the unmarked sequents form the active *fringe* of the database.
5. Stop when the goal sequent is matched, using the conditions of the completeness theorem (theorem 3.21). Otherwise, if no rules apply, abort the search procedure.

The particular saturating search strategy we use is the OTTER loop [60]. The procedure maintains two continually updated sequent databases.

Definition 4.17 (Sequent databases)**.**

1. *The* ***kept sequents*** *database (often referred to as the* set of support *in the literature [35]) contains new sequents that have not been subsumed, but are not yet being considered for rule applications.*
2. *The* ***active sequents*** *database that contains all sequents that should be considered for rule applications.*

At the start of each round of the OTTER loop, a sequent s is selected (and removed) from the kept sequents database and inserted into the active sequents database. This process sometimes goes by the name of "activation". Subsequently, all specialised rules are matched against s as the first premiss, and if any matches succeed, then the remaining premisses of those rules are matched against all sequents in the active sequents database. This is repeated until all the premisses of rules that have successful matches are satisfied and conclusions produced from these rules. The collection of conclusion sequents are then compared against all past sequents that were inserted into the kept database, and those sequents that are not subsumed are inserted into the kept database.

A great deal more can be said about the details of the implementation of the OTTER loop, but we will instead delay the presentation of the engineering details until after we have presented the focusing calculus. Much of the novelty of our variant of the OTTER loop comes from the the way we handle rules with several premisses. For more details, please see chapter 6. Completeness of the OTTER loop is a well known proerty [60], requiring only the following properties of the implementation.

Property 4.18 (Fair selection)**.** *Every sequent in the kept sequents database is eventually selected for insertion into the active sequents database.*

Property 4.19 (Fair application)**.** *If a rule can be fully satisfied by sequents in the active sequents database (in any order), then the conclusion of this rule is eventually considered for insertion into the kept sequents database.*

4.2 Proof extraction

As stated in the introduction (chapter 1), a main design goal of this work is to produce provers that are able to certify their proofs, i.e., produce independently verifiable proof objects. We have already seen a syntax for backward sequent derivations in chapter 2. This syntax used explicit variables for the hypotheses, which made extracting a natural deduction proof object from them a fairly trivial process (sec. 2.2.1).

However, in the forward direction we have no notion of hypothesis variables. In fact, multiple occurrences of a resource are abbreviated into a single label with an associated multiplicity, and the matching condition simply subtracts from the multiplicity of these resources. Thus, in the forward direction we lack a way to refer to any particular hypothesis, but must instead settle for the much weaker form of just referring to the label itself.

Fortunately, in the propositional case, this is enough information to reconstruct a natural deduction proof deterministically. The key observation is that whenever a resource needs to be matched in a premiss, it is sufficient to simply select the first resource that matches. This gives us a backward derivation from which the natural deduction proof object can be extracted as in sec. 2.2.1.

Syntax of forward derivations Forward derivations, written as $\mathcal{D}_f$, have the following syntactic forms with the obvious correspondences

$$\text{init}(p) \qquad \text{copy}(\mathcal{D}_f, l \mathbin{\#} A)$$

$$\otimes R(\mathcal{D}_f, \mathcal{D}'_f \,;\, r \mathbin{\#} A \otimes B) \qquad \otimes L(\mathcal{D}_f, l_A \mathbin{\#} A, l_B \mathbin{\#} B \,;\, l \mathbin{\#} A \otimes B) \qquad \mathbf{1}R(r \mathbin{\#} \mathbf{1}) \qquad \mathbf{1}L(\mathcal{D}_f \,;\, l \mathbin{\#} \mathbf{1})$$

$$\multimap R(\mathcal{D}_f, l \mathbin{\#} A \,;\, r \mathbin{\#} A \multimap B) \qquad \multimap L(\mathcal{D}_f, (\mathcal{D}'_f, l \mathbin{\#} B) \,;\, l' \mathbin{\#} A \multimap B)$$

$$\& R(\mathcal{D}_f, \mathcal{D}'_f \,;\, r \mathbin{\#} A \mathbin{\&} B) \qquad \& L_i(\mathcal{D}_f, l \mathbin{\#} A_i \,;\, l' \mathbin{\#} A_1 \mathbin{\&} A_2) \qquad \top R(r \mathbin{\#} \top)$$

$$\oplus R_i(\mathcal{D}_f, r \mathbin{\#} A_i \,;\, r' \mathbin{\#} A_1 \oplus A_2) \qquad \oplus L((\mathcal{D}_f, l_A \mathbin{\#} A), (\mathcal{D}'_f, l_B \mathbin{\#} B) \,;\, l \mathbin{\#} A \oplus B) \qquad \mathbf{0}L(l \mathbin{\#} \mathbf{0})$$

$$!R(\mathcal{D}_f \,;\, r \mathbin{\#}\, !A) \qquad !L(\mathcal{D}_f, l \mathbin{\#} A \,;\, l' \mathbin{\#}\, !A)$$

Definition 4.20 (Forward labelled sequent)**.** *A forward labelled sequent is like a forward sequent with every proposition labelled. That is, it has the following shape:*

$$\underbrace{u_1 :: l_{u1} \mathbin{\#} A_1, \ldots, u_m :: l_{um} \mathbin{\#} A_m}_{\Gamma_l} \quad ; \quad [\underbrace{v_1 : l_{l1} \mathbin{\#} B_1, \ldots, v_n :: l_{ln} \mathbin{\#} B_n}_{\Delta_l}]_w \quad \longrightarrow \quad \underbrace{\begin{cases} r \mathbin{\#} C \\ \cdot \end{cases}}_{\gamma_l}$$

Definition 4.21 (Forward derivations to backward derivations)**.** *We define the translation of a forward derivation of the sequent representation $\mathcal{D}_r :: \Gamma_r \,;\, [\Delta_r]_w \longrightarrow \gamma_r$ to a forward derivation of a labelled sequent $\mathcal{D}_f :: \Gamma_l \,;\, [\Delta_l]_w \longrightarrow \gamma_l$, written as $\mathcal{D}_r :: \Gamma_r \,;\, [\Delta_r]_w \longrightarrow \gamma_r \rightsquigarrow \mathcal{D}_f :: \Gamma_l \,;\, [\Delta_l]_w \longrightarrow \gamma_l$ by means of the suitable deterministic rules. Some typical examples of such rules:*

$$\frac{u \textit{ fresh}}{\textit{init}(p) :: \cdot \,;\, [p^1 \mathbin{\#} p]_0 \longrightarrow p \rightsquigarrow \textit{init}(u{:}p) :: \cdot \,;\, [u{:}p \mathbin{\#} p]_0 \longrightarrow p} \rightsquigarrow \text{init}$$

$$\frac{\mathcal{D}_r :: \Gamma_r \,;\, [\Delta_r, l^k \mathbin{\#} A]_w \longrightarrow \gamma_r \rightsquigarrow \mathcal{D}_f :: \Gamma_l \,;\, [\Delta_l, v : A]_w \longrightarrow \gamma_l \quad u \textit{ fresh}}{\textit{copy}(\mathcal{D}_r, l \mathbin{\#} A) :: \Gamma_r \sqcup l \mathbin{\#} A \,;\, [\Delta_r, l^{k-1} \mathbin{\#} A]_w \longrightarrow \gamma_r \rightsquigarrow \textit{copy}(v.\, \mathcal{D}_f \,;\, u : A) :: \Gamma_l, u{:}A \,;\, [\Delta_l]_w \longrightarrow \gamma_l} \rightsquigarrow \text{copy}$$

$$\frac{\begin{array}{c} \mathcal{D}_r :: \Gamma_r \,;\, [\Delta_r]_w \longrightarrow A \rightsquigarrow \mathcal{D}_f :: \Gamma_l \,;\, [\Delta_l]_w \longrightarrow A \\ \mathcal{D}'_r :: \Gamma'_r \,;\, [\Delta'_r]_{w'} \longrightarrow B \rightsquigarrow \mathcal{D}'_l :: \Gamma'_l \,;\, [\Delta'_l]_{w'} \longrightarrow B \end{array}}{\begin{array}{c} \otimes R(\mathcal{D}_r, \mathcal{D}'_r \,;\, r \mathbin{\#} A \otimes B) :: \Gamma_r \sqcup \Gamma'_r \,;\, [\Delta_r, \Delta'_r]_{w \vee w'} \longrightarrow A \otimes B \\ \rightsquigarrow \otimes R(\mathcal{D}_l, \mathcal{D}'_l \,;\, A \otimes B) :: \Gamma_l \sqcup \Gamma'_l \,;\, [\Delta_l, \Delta'_l]_{w \vee w'} \longrightarrow A \otimes B \end{array}} \rightsquigarrow \otimes R$$

Given a forward labelled sequent s_f, define $\wr s_f \int$ as the forward sequent produced by erasing all the subformula labels. We then have:

Theorem 4.22 (Soundness of proof extraction)**.** *If* $\mathcal{D}_r :: \Gamma_r \,; [\Delta_r]_w \longrightarrow \gamma_r$ *and* $\mathcal{D}_r :: \Gamma_r \,; [\Delta_r]_w \longrightarrow \gamma_r \rightsquigarrow \mathcal{D}_f :: \Gamma_l \,; [\Delta_l]_w \longrightarrow \gamma_l$*, then* $\mathcal{D}_f :: \wr \Gamma_l \,; [\Delta_l]_w \longrightarrow \gamma_l \int$.

Proof. By induction on the structure of $\mathcal{D}_f$ and definition 4.21. □

Once we are able to extract a standard forward derivation of a forward sequent, we simply appeal to theorem 3.9 to extract the corresponding backward derivation, then definition 2.24 to extract the corresponding natural deduction proof from it.

The actual implementation of the proof extraction procedure does not take this long tour through a number of sequent calculi, but instead directly extracts the natural deduction proof from the forward sequent calculus. This process is not formalised here because it amounts to performing the steps in definition 2.24 in tandem with the proof of theorem 3.9, which, although tedious, is a straightforward process.

4.3 Historical review

Historically, the inverse method for classical (non-linear) logic owes its development to Maslov [73]. Subsequently, Voronkov [116], Mints [84], and more recently Tammet [108, 110] have adapted it for non-classical and intuitionistic logics, though not for linear logic. Mints [83] has investigated resolution calculi for classical linear logic, but his methods don't have an immediate application to the inverse method. Many elements of the inverse method are described well in the handbook article on this topic [35]

The material in this chapter has been published in a more preliminary form in [28].

Chapter summary *In this chapter we have presented the inverse method algorithm that accompanies the forward calculus of the previous chapter. The key concept in the inverse method is to use the subformula property to construct specialised rules for labelled subformulas. A full small-step prover can in fact be constructed from this outline, and we present such a prover (named* LIPF*) in chapter 7.*

In the next chapter the forward sequent calculus will be extended to the full first-order setting. The updates to the inverse method in the presence of quantifiers will be discussed from sec. 5.4 onwards.

Chapter 5

First Order Quantification

In this chapter we will consider the problem of extending the propositional forward sequent calculus and the inverse method implementation based on that with first-order quantifiers. The primary problem in the presence of quantifiers is the need to deal with term variables for which syntactic equality has to be relaxed to unifiability. In other words, equalities are not manifestly present, but arise as a result of computation of unifiers.

We follow here the "recipe" laid out in the chapter on the inverse method in the Handbook article [35]. First we present a ground forward calculus with no free variables. This calculus will not be implementable because it will have fully instantiated initial sequents that cannot be computed directly as subformulas of the goal sequent. This calculus will be shown to be sound and complete with respect to the first-order backward sequent calculus of section 2.1.4. Next, a lifted version of this calculus will be constructed which will have instantiable variables. We will then show that any derivation in the ground calculus is merely an instance of a corresponding derivation in the lifted calculus. The completeness theorem will then be in terms of not only finding a possibly stronger form of the goal sequent, but also a possibly more general sequent.

When applying the "recipe" for linear logic, we have to consider several complications having to do with linearity. The most important of these complications is that for binary additive rules the two input linear contexts have to be compared not just for equality but for unifiability. The problem amounts to unifying multisets of predicates. There are several ways to solve this problem; we proceed by observing that uniting two contexts is a kind of factoring of common sub-contexts. We therefore present this context unification directly in terms of an algorithm for contraction.

The rest of this chapter is structured as follows. In sec. 5.1, we extend the subformula properties of earlier chapters to the first-order case, and introduce the concept of *free subformula*. In sec. 5.2 we present the ground forward calculus and prove it sound and complete with respect to the backward calculus of sec. 2.1.4. In sec. 5.3 we lift this calculus to free variables and prove the important completeness theorem (thm. 5.23). We then switch to implementation details. In sec. 5.4 we show how we represent sequents in this lifted forward calculus, and in sec. 5.5 describe how we implement subsumption and indexing. The chapter concludes with a discussion of the modifications to the inverse method of chapter 4 to memoise partially applied multi-premiss rules in sec. 5.6.

5.1 Quantification and the subformula property

As stated in section 2.1.4, we extend the language of propositions with first-order quantification over a language of untyped *terms*

$$\text{(terms)} \qquad s, t, \ldots \quad ::= \quad x \quad | \quad f(t_1, t_2, \ldots, t_n)$$

where x ranges over a countably infinite set of *variables*, and f over a collection of *function symbols*. We also extend the language of propositions with universal ($\forall$) and existential ($\exists$) quantification over these terms. Thus, we must now extend the subformula relation to handle quantification. We adopt the easy extensions of the definitions of (partial) decoration (definitions 4.1, 4.2, and 4.3).

Definition 5.1 (First-order decorated subformula relation)**.** *We extend the definition of signed decorated subformula relation for the propositional case (defn. 4.4) with cases for the first-order quantifiers.*

$$[a/x]A^+ \leq (\forall x.A)^+ \qquad\qquad [t/x]A^- \leq (\forall x.A)^-$$
$$[t/x]A^+ \leq (\exists x.A)^+ \qquad\qquad [a/x]A^- \leq (\exists x.A)^-$$

where t ranges over arbitrary terms, and a ranges over parameters, i.e., uninterpreted global constants. We also adopt the standard abuses of notation as in defn. 4.4.

The decorated subsequent relation is, once again, adopted from the propositional case (defn. 4.5). Recall (defn. 4.1) that a subscripted $_!$ indicates that the subformula may be copied into the unrestricted context.

Theorem 5.2 (Subformula property). *If* $\Gamma' \mathbin{;} \Delta' \Longrightarrow C'$ *appears in a proof of* $\Gamma \mathbin{;} \Delta \Longrightarrow C$*, then:*

$$\Gamma'^{-}_{!} \cup \Delta'^{-}_{\cdot} \cup C'^{+}_{\cdot} \le \Gamma^{-}_{!} \cup \Delta^{-}_{\cdot} \cup C^{+}_{\cdot}$$

Proof. The proof is by induction on the structure of $\mathcal{D} :: \Gamma \mathbin{;} \Delta \Longrightarrow C$, as for the proof of thm. 5.2, but we now have to account for the cases of the first-order quantifiers. The induction hypothesis is applicable whenever we have a smaller sequent in the subsequent relation. To illustrate, here is the case of $\forall R$.

$$\frac{\Gamma \mathbin{;} \Delta \Longrightarrow [a/x]A}{\Gamma \mathbin{;} \Delta \Longrightarrow \forall x.A}\ \forall R^{a}$$

Let a parameter a be given. Then $[a/x]A \le \forall x.A$, so using the induction hypothesis every sequent in the proof of $\Gamma \mathbin{;} \Delta \Longrightarrow [a/x]A$ contains subformulas of $\Gamma^{-}_{!} \cup \Delta^{-}_{\cdot} \cup ([a/x]A)^{+}_{\cdot}$, which is a subset of the subformulas of $\Gamma^{-}_{!} \cup \Delta^{-}_{\cdot} \cup (\forall x.\, A)^{+}_{\cdot}$. A similar argument is used for the other quantifier rules. $\square$

To simplify matters, for the rest of this work we will adopt the convention of considering only *rectified* goal sequents [35].

Definition 5.3. *A proposition A is* rectified *if*

(a) all first-order quantifiers in A bind a different variable; and
(b) every bound variable in A has no free occurrence in A.

A collection of propositions is rectified if all members of the collection are rectified, and furthermore all bound variables are distinct across the entire collection. A sequent is rectified if the collection of propositions on the left and right of the sequent arrow is rectified.

Rectification allows us to define a second kind of subformula, sometimes called *free subformula* [35], to refer to the sub-units of a given proposition syntactically. This subformula relation has no associated subformula property, but it is necessary in the definition of lifted forward sequent calculus.

Definition 5.4 (Free subformulas)**.** *The free subformula relation $\ll$ on propositions is the reflexive-transitive closure of the following rules:*

$$A \ll (A * B) \quad B \ll (A * B) \quad A \ll {!A} \quad A \ll \forall x.\, A \quad A \ll \exists x.\, A \quad \ldots * \in \{\otimes, \&, \oplus, \multimap\}$$

5.2 Ground forward sequent calculus

We begin first by constructing a forward calculus with no instantiable free variables or unification. That is to say, in this calculus all equalities will be manifestly syntactic and no equalities can be induced in the computations involved in rule-application. This will not be an implementable calculus because there will (generally) be infinitely many initial sequents because the subformula relation in the presence of quantifiers is infinite. But, it is a necessary step in establishing the completeness of a lifted calculus with free variables in section 5.3.

We adopt the same sequent structure in the first-order setting as in sec. 3.2. Most of the rules of the calculus are presentationally identical to the rules in that section, so we shall list here only the differences. First the new additions – the quantifier rules.

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow [a/x]A}{\Gamma\,;[\Delta]_w \longrightarrow \forall x.\, A}\ \forall R^a \qquad \frac{\Gamma\,;[\Delta, [t/x]A]_w \longrightarrow \gamma}{\Gamma\,;[\Delta, \forall x.\, A]_w \longrightarrow \gamma}\ \forall L$$

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow [t/x]A}{\Gamma\,;[\Delta]_w \longrightarrow \exists x.\, A}\ \exists R \qquad \frac{\Gamma\,;[\Delta, [a/x]A]_w \longrightarrow \gamma}{\Gamma\,;[\Delta, \exists x.\, A]_w \longrightarrow \gamma}\ \exists L^a$$

As usual, the superscript ^{a} in $\forall R$ and $\exists L$ denote that the parameter a does not occur in the conclusion of the sequent.

Initial sequents need to account for the term arguments to the atomic predicates.

$$\frac{}{\cdot\,;[p(\vec{t})]_0 \longrightarrow p(\vec{t})}\ \text{init}$$

No search procedure can generate these sequents a-priori as $\vec{t}$ are terms produced by the subformula relation. Because there are infinitely many terms, there are infinitely many subformulas of $(\exists x.\, p(x))^+_\cdot$, for example, each of the form $p(t)$ for some term t, and therefore infinitely many initial sequents that can derive $\cdot\,;[\exists x.\, p(x)]_0 \longrightarrow \exists x.\, p(x)$. The "init" rule should therefore be read as assigning arguments—restricted by the subformula relation, of course—to the atomic predicates for which the left and right are equal.

The multiplicative and exponential rules are the same as in sec. 3.2. For the additive rules, the notion of additive composition must be extended in the first-order setting. It is certainly possible to give a definition such as the following even in the first-order setting:

$$[\Delta]_w + [\Delta']_{w'} = \begin{cases} [\Delta]_0 & \text{if } w = w' = 0 \text{ and } \Delta = \Delta' \\ [\Delta]_0 & \text{if } w = 0, w' = 1 \text{ and } \Delta' \subseteq \Delta \\ [\Delta']_0 & \text{if } w = 1, w' = 0 \text{ and } \Delta \subseteq \Delta' \\ [\Delta \sqcup \Delta']_1 & \text{if } w = w' = 1 \end{cases}$$

When lifted to free variables, however, this analysis breaks down because we have to deal with unifiability rather than equality of propositions. The test $\Delta = \Delta'$ might be provable in different ways using incompatible substitutions, and there will generally be many possible ways to compute $\Delta \sqcup \Delta'$. Foreshadowing these complications, we abandon this declarative presentation and instead present the *compatibility* of contexts as an algorithm; this will also ease the proof of completeness of the lifted calculus.

Definition 5.5. Additive contraction *of two linear contexts* $[\Delta_1]_{w_1}$ *and* $[\Delta_2]_{w_2}$ *to produce a third linear context* $[\Delta]_w$*, written* $[\Delta_1]_{w_1} + [\Delta_2]_{w_2} \rightsquigarrow [\Delta]_w$*, is governed by the following rules.*

$$\frac{}{[\cdot]_0 + [\cdot]_0 \rightsquigarrow [\cdot]_0} \rightsquigarrow_{00} \qquad \frac{}{[\Delta]_0 + [\cdot]_1 \rightsquigarrow [\Delta]_0} \rightsquigarrow_{01} \qquad \frac{}{[\cdot]_0 + [\Delta]_1 \rightsquigarrow [\Delta]_0} \rightsquigarrow_{10}$$

$$\frac{}{[\Delta]_1 + [\Delta']_1 \rightsquigarrow [\Delta, \Delta']_1} \rightsquigarrow_{11} \qquad \frac{[\Delta_1]_{w_1} + [\Delta_2]_{w_2} \rightsquigarrow [\Delta, A]_w}{[\Delta_1, A]_{w_1} + [\Delta_2, A]_{w_1} \rightsquigarrow [\Delta, A]_w} \rightsquigarrow$$

The contexts $[\Delta_1]_{w_1}$ *and* $[\Delta_2]_{w_2}$ *are* additively compatible *if they can be additively contracted.*

Note that $\rightsquigarrow$ is non-deterministic because the fourth and fifth rules overlap.

Lemma 5.6 (Simple properties)**.**

1. *If* $[\Delta_1]_{w_1} + [\Delta_2]_{w_2} \rightsquigarrow [\Delta]_w$*, then* $[\Delta_2]_{w_2} + [\Delta_1]_{w_1} \rightsquigarrow [\Delta]_w$.
2. *If* $[\Delta_1]_1 + [\Delta_2]_0 \rightsquigarrow [\Delta]_w$*, then* $\Delta_1 \subseteq \Delta_2$.
3. *If* $[\Delta_1]_0 + [\Delta_2]_1 \rightsquigarrow [\Delta]_w$*, then* $\Delta_2 \subseteq \Delta_1$.

Proof. Each property can be proved by simple induction on the derivation of $\rightsquigarrow$. □

The result of additive contraction of two weak contexts is not necessarily equal to the least upper bound ($\sqcup$), but it certainly contains the least upper bound.

Lemma 5.7. *If* $[\Delta_1]_1 + [\Delta_2]_1 \rightsquigarrow [\Delta]_1$*, then* $\Delta_1 \sqcup \Delta_2 \subseteq \Delta \subseteq \Delta_1, \Delta_2$*.*

Proof. Use the "$\rightsquigarrow$" rule to contract the elements of $(\Delta_1, \Delta_2) \setminus \Delta$. □

We write the additive rules using the additive contraction judgement as follows (for the example of $\&R$):

$$\frac{\Gamma_1 \,;\, [\Delta_1]_{w_1} \longrightarrow A \quad \Gamma_2 \,;\, [\Delta_2]_{w_2} \longrightarrow B \quad [\Delta_1]_{w_1} + [\Delta_2]_{w_2} \rightsquigarrow [\Delta]_w}{\Gamma_1, \Gamma_2 \,;\, [\Delta]_w \longrightarrow A \mathbin{\&} B} \ \&R$$

The full complement of rules is given in figure 5.1. In establishing soundness, we extend the $(-)^o$ mapping from forward to backward derivations to account for the quantifier rules.

Definition 5.8 (Forward derivations to backward derivations)**.** *The mapping from forward to backward derivations (defn. 3.1) is extended to handle the quantifier rules in the usual fashion.*

Theorem 5.9 (Soundness of forward derivations)**.**

1. *If* $\mathcal{D}_f :: \Gamma \,;\, [\Delta]_0 \longrightarrow C$*, then* $(\mathcal{D}_f)^o :: \Gamma \,;\, \Delta \Longrightarrow C$*.*
2. *If* $\mathcal{D}_f :: \Gamma \,;\, [\Delta]_1 \longrightarrow \gamma$*, then for any* $\Delta' \supseteq \Delta$ *and* $C \supseteq \gamma$*,* $(\mathcal{D}_f)^o :: \Gamma \,;\, \Delta' \Longrightarrow C$*.*

Proof. By induction on the structure of $\mathcal{D}_f$. The proof is a natural extension of the proof of theorem 3.9. The major departures are for the quantifier rules for which we need to allow the induction hypothesis to be applicable for α-renaming of the free variables of smaller sequents, and for the uses of the contraction judgement. The following are some representative cases.

Case $\otimes R$:

$$\frac{\mathcal{D}_{f1} :: \Gamma_1 \,;\, [\Delta_1]_{w_1} \longrightarrow A \quad \mathcal{D}_{f2} :: \Gamma_2 \,;\, [\Delta_2]_{w_2} \longrightarrow B}{\otimes R(\mathcal{D}_{f1}, \mathcal{D}_{f2} \,;\, A \otimes B) :: \Gamma_1 \sqcup \Gamma_2 \,;\, [\Delta_1, \Delta_2]_{w_1 \vee w_2} \longrightarrow A \otimes B} \ \otimes R$$

Note that $(\otimes R(\mathcal{D}_{f1}, \mathcal{D}_{f2} \,;\, A \otimes B))^o = \otimes R((\mathcal{D}_{f1})^o, (\mathcal{D}_{f2})^o \,;\, A \otimes B)$.

Subcase $w_1 = w_2 = 0$. In this case,

$(\mathcal{D}_{f1})^o :: \Gamma_1 \,;\, \Delta_1 \Longrightarrow A$	i.h.
$(\mathcal{D}_{f1})^o :: \Gamma_1 \sqcup \Gamma_2 \,;\, \Delta_1 \Longrightarrow A$	lem. 5.6 and weakening
$(\mathcal{D}_{f1})^o :: \Gamma_1 \sqcup \Gamma_2 \,;\, \Delta_2 \Longrightarrow B$	similarly
$\otimes R((\mathcal{D}_{f1})^o, (\mathcal{D}_{f2})^o \,;\, A \otimes B) :: \Gamma_1 \sqcup \Gamma_2 \,;\, \Delta_1, \Delta_2 \Longrightarrow A \otimes B$	$\otimes R$

Judgemental

$$\frac{}{\cdot\,;[p(\vec{t})]_0 \longrightarrow p(\vec{t})}\ \text{init} \qquad \frac{\Gamma\,;[\Delta, A]_w \longrightarrow \gamma}{\Gamma, A\,;[\Delta]_w \longrightarrow \gamma}\ \text{copy} \qquad \frac{\Gamma, A, A\,;[\Delta]_w \longrightarrow \gamma}{\Gamma, A\,;[\Delta]_w \longrightarrow \gamma}\ \text{factor}$$

Multiplicative

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow A \quad \Gamma'\,;[\Delta']_{w'} \longrightarrow B}{\Gamma, \Gamma'\,;[\Delta, \Delta']_{w \vee w'} \longrightarrow A \otimes B}\ {\otimes}R \qquad \frac{\Gamma\,;[\Delta, A, B]_w \longrightarrow \gamma}{\Gamma\,;[\Delta, A \otimes B]_w \longrightarrow \gamma}\ {\otimes}L \qquad \frac{\Gamma\,;[\Delta, A_i]_1 \longrightarrow \gamma}{\Gamma\,;[\Delta, A_1 \otimes A_2]_1 \longrightarrow \gamma}\ {\otimes}L'$$

$$\frac{}{\cdot\,;[\cdot]_0 \longrightarrow \mathbf{1}}\ \mathbf{1}R \qquad \frac{\Gamma\,;[\Delta]_w \longrightarrow \gamma}{\Gamma\,;[\Delta, \mathbf{1}]_w \longrightarrow \gamma}\ \mathbf{1}L$$

$$\frac{\Gamma\,;[\Delta, A]_w \longrightarrow \gamma \quad B \supseteq \gamma}{\Gamma\,;[\Delta]_w \longrightarrow A \multimap B}\ {\multimap}R \qquad \frac{\Gamma\,;[\Delta]_1 \longrightarrow B}{\Gamma\,;[\Delta]_1 \longrightarrow A \multimap B}\ {\multimap}R'$$

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow A \quad \Gamma'\,;[\Delta', B]_{w'} \longrightarrow C}{\Gamma, \Gamma'\,;[\Delta, \Delta', A \multimap B]_{w \vee w'} \longrightarrow C}\ {\multimap}L$$

Additive

$$\frac{\begin{array}{l}\Gamma\,;[\Delta]_w \longrightarrow A \\ \Gamma'\,;[\Delta']_{w'} \longrightarrow B \quad [\Delta]_w + [\Delta']_{w'} \leadsto [\Delta'']_{w''}\end{array}}{\Gamma, \Gamma'\,;[\Delta'']_{w''} \longrightarrow A \,\&\, B}\ \&R \qquad \frac{\Gamma\,;[\Delta, A_i]_w \longrightarrow \gamma}{\Gamma\,;[\Delta, A_1 \,\&\, A_2]_w \longrightarrow \gamma}\ \&L_i$$

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow A_i}{\Gamma\,;[\Delta]_w \longrightarrow A_1 \oplus A_2}\ {\oplus}R_i$$

$$\frac{\begin{array}{l}\Gamma\,;[\Delta, A]_w \longrightarrow \gamma \\ \Gamma'\,;[\Delta']_{w'} \longrightarrow \gamma' \quad [\Delta]_w + [\Delta']_{w'} \leadsto [\Delta'']_{w''}\end{array}}{\Gamma, \Gamma'\,;[\Delta'', A \oplus B]_{w''} \longrightarrow \gamma \cup \gamma'}\ {\oplus}L \qquad \frac{\begin{array}{l}\Gamma\,;[\Delta]_w \longrightarrow \gamma \\ \Gamma'\,;[\Delta', B]_{w'} \longrightarrow \gamma' \quad [\Delta]_w + [\Delta']_{w'} \leadsto [\Delta'']_{w''}\end{array}}{\Gamma, \Gamma'\,;[\Delta'', A \oplus B]_{w''} \longrightarrow \gamma \cup \gamma'}\ {\oplus}L'$$

$$\frac{}{\cdot\,;[\cdot]_1 \longrightarrow \top}\ {\top}R \qquad \frac{}{\cdot\,;[\mathbf{0}]_1 \longrightarrow \cdot}\ \mathbf{0}L$$

Exponential

$$\frac{\Gamma\,;[\cdot]_w \longrightarrow A}{\Gamma\,;[\cdot]_0 \longrightarrow\, !A}\ !R \qquad \frac{\Gamma\,;[\Delta]_w \longrightarrow \gamma}{\Gamma \backslash A\,;[\Delta, !A]_w \longrightarrow \gamma}\ !L$$

Quantifiers

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow [a/x]A}{\Gamma\,;[\Delta]_w \longrightarrow \forall x.\, A}\ \forall R^a \qquad \frac{\Gamma\,;[\Delta, [t/x]A]_w \longrightarrow \gamma}{\Gamma\,;[\Delta, \forall x.\, A]_w \longrightarrow \gamma}\ \forall L$$

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow [t/x]A}{\Gamma\,;[\Delta]_w \longrightarrow \exists x.\, A}\ \exists R \qquad \frac{\Gamma\,;[\Delta, [a/x]A]_w \longrightarrow \gamma}{\Gamma\,;[\Delta, \exists x.\, A]_w \longrightarrow \gamma}\ \exists L^a$$

Figure 5.1: Rules for the ground forward first-order calculus

Subcase $w_1 = 1$. Then, $w_1 \vee w_2 = 1$, so let $\Delta' \supseteq \Delta_1, \Delta_2 = \Delta$ be given.

$$\begin{array}{lr}
(\mathcal{D}_{f1})^o :: \Gamma_1 \,;\, \Delta' \backslash \Delta_2 \Longrightarrow A & \text{i.h.} \\
(\mathcal{D}_{f1})^o :: \Gamma_1 \sqcup \Gamma_2 \,;\, \Delta' \backslash \Delta_2 \Longrightarrow A & \text{lem. 5.6 and weakening} \\
(\mathcal{D}_{f2})^o :: \Gamma_1 \sqcup \Gamma_2 \,;\, \Delta_2 \Longrightarrow B & \text{similarly} \\
\otimes R((\mathcal{D}_{f1})^o, (\mathcal{D}_{f2})^o \,;\, A \otimes B) :: \Gamma_1 \sqcup \Gamma_2 \,;\, \Delta' \Longrightarrow A \otimes B & \otimes R
\end{array}$$

Case $\forall R$:

$$\frac{\mathcal{D}_f :: \Gamma \,;\, [\Delta]_w \longrightarrow [a/x]A}{\forall R(\mathcal{D}_f, a \,;\, \forall x.\, A) :: \Gamma \,;\, [\Delta]_w \longrightarrow \forall x.\, A}\ \forall R^a$$

Consider the case of $w = 0$. Let a parameter b be given, that is to say, let b be such that it does not occur in the conclusion $\Gamma \,;\, [\Delta]_w \longrightarrow \forall x.\, A$. Because the induction hypothesis is applicable for all α-renaming of the given sequent, we have.

$$\begin{array}{lr}
[b/a](\mathcal{D}_f)^o :: \Gamma \,;\, \Delta \Longrightarrow [b/x]A & \text{i.h.} \\
\forall R([b/a](\mathcal{D}_f)^o, b \,;\, \forall x.\, A) :: \Gamma \,;\, \Delta \Longrightarrow \forall x. A & \forall R
\end{array}$$

We then note that derivations are equal up to α-renaming, so $(\forall R(\mathcal{D}_f, a \,;\, \forall x.\, A))^o = \forall R((\mathcal{D}_f)^o, a \,;\, \forall x. A)$. The case of $w = 1$ is similar.

Case $\exists R$:

$$\frac{\mathcal{D}_f :: \Gamma \,;\, [\Delta]_w \longrightarrow [t/x]A}{\exists R(\mathcal{D}_f, t \,;\, \exists x.\, A) :: \Gamma \,;\, [\Delta]_w \longrightarrow \exists x.\, A}\ \exists R$$

Again, let us take the case of $w = 0$. We have:

$$\begin{array}{lr}
(\mathcal{D}_F)^o :: \Gamma \,;\, \Delta \Longrightarrow [t/x]A & \\
\exists R((\mathcal{D}_f)^o, t \,;\, \exists x.\, A) :: \Gamma \,;\, \Delta \Longrightarrow \exists x.\, A & \exists R
\end{array}$$

We then note that $(\exists R(\mathcal{D}_f, t \,;\, \exists x. A))^o = \exists R((\mathcal{D}_f)^o, t \,;\, \exists x. A)$. The case of $w = 1$ is similar.

□

For the completeness theorem we reprise theorem 3.11, but extended with the new rules.

Theorem 5.10 (Completeness of forward derivations).
Suppose $\Gamma \,;\, \Delta \Longrightarrow C$. *Then,*

(a) either $\Gamma' \,;\, [\Delta]_0 \longrightarrow C$,

(b) or $\Gamma' \,;\, [\Delta']_1 \longrightarrow \gamma$

for some $\Gamma' \subseteq \Gamma$, $\Delta' \subseteq \Delta$ *and* $\gamma \subseteq C$.

Proof. Like in the proof of thm. 3.11, we prove this by induction on the structure of the derivation $\mathcal{D}_b :: \Gamma\,;\Delta \Longrightarrow C$. The following are the representative cases for the exponentials.

Case $\forall R$:

$$\frac{\mathcal{D}_b :: \Gamma\,;\Delta \Longrightarrow [a/x]A}{\forall R(\mathcal{D}_b, a\,;\forall x.\,A) :: \Gamma\,;\Delta \Longrightarrow \forall x.\,A}\ \forall R^a$$

As usual, there are two sub-cases for the induction hypothesis on the premiss. In the first case, we have a strong sequent:

$\Gamma'\,;[\Delta]_0 \longrightarrow [a/x]A$ for some $\Gamma' \subseteq \Gamma$ — case of i.h.

$\Gamma'\,;[\Delta]_0 \longrightarrow \forall x.\,A$ — $\forall R$ because a doesn't occur in Γ' and Δ.

In the other case we have a weak sequent. If the right hand side is empty, then we are already done, so assume it is non-empty.

$\Gamma'\,;[\Delta']_1 \longrightarrow [a/x]A$ for some $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$ — case of i.h.

$\Gamma'\,;[\Delta']_1 \longrightarrow \forall x.\,A$ — $\forall R$

The cases of $\exists R$, $\forall L$ and $\exists L$ are very similar. □

5.3 Lifting to free variables

The calculus of the previous section uses only ground initial sequents, which is impossible for an implementation of the forward calculus. Continuing with the "recipe" from [35], in this section we present a lifted version of the calculus with explicit unification. We begin, as usual, by fixing a goal sequent $\Gamma_g\,;[\Delta_g]_w \longrightarrow C_g$ and considering only the free subformulas of this goal. In the presentation, the quantified propositions are silently α-renamed as necessary. In this calculus, every proposition on the left and right is accompanied by a substitution for some of its parameters or free term variables. These substitutions are built according the following grammar:

(substitutions)	$\sigma ::= \epsilon$	(identity)
	$\mid\ \sigma, a_1/a_2$	(param-subst)
	$\mid\ \sigma, t/x$	(term-subst)

The parameters and term variables being substituted for are all distinct in a substitution. We write $A[\sigma]$ (resp. $t[\sigma]$) for the application of the substitution σ to the proposition A (resp. term t) with free parameters or term variables. Sequents in the free calculus contain free subformula/substitution pairs, written $A \cdot \sigma$. The sequencing of σ followed by ξ, written $\sigma\xi$, has the property $A[\sigma\xi] = (A[\sigma])[\xi]$. The composition of θ with every substitution in a zone Γ or Δ (now containing formula/substitution pairs) is written $\Gamma\theta$ or $\Delta\theta$. The identity substitution ϵ will be elided unless absolutely necessary for clarity.

A minor novel aspect of our formulation is that we distinguish parameters (which can be substituted only for other parameters) and variables (for which we can substitute arbitrary terms, including parameters). The distinction arises from the notion of subformula, since positive universal and negative existential formulas can only ever be instantiated with parameters in a cut-free backward sequent derivation. We achieve this by syntactically distinguishing positive universal and negative existential subformulas in the goal sequent to bind parameters, i.e., of the form $\forall a.\ A$ and $\exists a.\ A$. The free subformulas of the goal sequent will thus have parameters in the expected positions.

This sharpening sometimes removes unreachable initial sequents from consideration. Where the distinction between term variables and parameters is not relevant, we shall write them using lowercase Greek letters α, β, etc.

Definition 5.11 (Standard definitions)**.**

1. *The* domain *of a substitution $\sigma = t_1/\alpha_1, \ldots, t_n/\alpha_n$, written* $\mathrm{dom}(\sigma)$*, is the set* $\{\alpha_1, \ldots, \alpha_n\}$.
2. *The* range *of a substitution σ, written* $\mathrm{rng}(\sigma)$*, is the set* $\{\alpha[\sigma] : \alpha \in \mathrm{dom}(\sigma)\}$.
3. *The* image *of a substitution σ, written* $\mathrm{img}(\sigma)$*, is* $\mathrm{vars}(\mathrm{rng}(\sigma))$.
4. *Given two substitutions $\sigma = s_1/\alpha_1, \ldots, s_m/\alpha_m$ and $\theta = t_1/\beta_1, \ldots, t_n/\beta_n$ with disjoint domains, their composition $\sigma\theta$ is the substitution $s_1[\theta]/\alpha_1, \ldots, s_n[\theta]/\alpha_m, t_1/\beta_1, \ldots, t_n/\beta_n$. If the domains of σ and θ are not disjoint, then σ is first restricted to* $\mathrm{dom}(\sigma) \setminus \mathrm{dom}(\theta)$.
5. *Given two substitutions $\sigma = s_1/\alpha_1, \ldots, s_m/\alpha_m$ and $\theta = t_1/\beta_1, \ldots, t_n/\beta_n$ for which* $\mathrm{dom}(\sigma) \cap \mathrm{dom}(\theta) = \emptyset$*, their* merge*, written $\sigma \uplus \theta$, is $s_1/\alpha_1, \ldots, s_m/\alpha_m, t_1/\beta_1, \ldots, t_n/\beta_n$.*
6. *A substitution σ* agrees with *a substitution θ on a set of parameter and term variables V if for every $\alpha \in V$, $\alpha[\sigma] = \alpha[\theta]$.*
7. *The* restriction *of a substitution σ to a set of parameter and term variables V, written $\sigma|_V$ is a substitution with domain* $\mathrm{dom}(\sigma) \cap V$ *that agrees with σ on V.*

The standard notion of most general unifier carries over in a straightforward way to this slightly more general setting.

Definition 5.12 (Most general unifiers).

1. *The most general unifier of t_1 and t_2, written* $\mathrm{mgu}(t_1, t_2)$*, is a unifier θ such that for any other unifier σ of t_1 and t_2, there is a ξ such that $\sigma = \theta\xi$.*
2. *The most general unifier of lists of terms $\vec{t_1}$ and $\vec{t_2}$ of equal length, written* $\mathrm{mgu}(\vec{t_1}, \vec{t_2})$*, equals* $\mathrm{mgu}(\star(\vec{t_1}), \star(\vec{t_2}))$ *where $\star$ is some function symbol not occuring in $\vec{t_1}$ and $\vec{t_2}$.*
3. *The most general unifier of σ and θ, written* $\mathrm{mgu}(\sigma, \theta)$*, equals* $\mathrm{mgu}(\vec{x}[\sigma], \vec{x}[\theta])$ *where* $\vec{x} = \mathrm{dom}(\sigma) \cup \mathrm{dom}(\theta)$*.*

Theorem 5.13 (Existence of most general unifiers). *If two terms t_1 and t_2 are unifiable, then* $\mathrm{mgu}(t_1, t_2)$ *exists.* □

We make the customary assumption that substitutions are idempotent.

Definition 5.14 (Idempotent substitutions). *A substitution σ is idempotent if $\sigma\sigma = \sigma$.*

A renaming substitution is a special case of an idempotent substitution for which each element of the domain is merely mapped to another variable.

Definition 5.15 (renaming substitution). *An idempotent substitution ρ is a renaming substitution if:*

(a) Every term variable $x \in$ $\mathrm{dom}(\rho)$ *is bound to a distinct term variable y; and*
(b) Every parameter $a \in$ $\mathrm{dom}(\rho)$ *is bound to a distinct parameter b.*

Renaming substitutions have a fairly straightforward characterisation.

Property 5.16 (Invertibility of renaming substitutions). *If ρ is a renaming substitution, then there exists a unique substitution ρ^{-1} for which $\rho\rho^{-1} = \rho^{-1}\rho = \epsilon$.* □

The definition of additive contraction needs to be lifted to free subformulas also.

Definition 5.17 (lifted additive contraction). *The* lifted additive contraction judgment, *written $[\Delta_1]_{w_1} + [\Delta_2]_{w_2} \rightsquigarrow [\Delta]_w, \xi$, takes as input $[\Delta_1]_{w_1}$ and $[\Delta_2]_{w_2}$ and produces a contracted context $[\Delta]_w$ and its corresponding substitution ξ. The rules for this judgment are as follows.*

$$\frac{}{[\cdot]_0 + [\cdot]_0 \rightsquigarrow [\cdot]_0, \epsilon} \rightsquigarrow_{00} \qquad \frac{}{[\cdot]_1 + [\Delta]_0 \rightsquigarrow [\Delta]_0, \epsilon} \rightsquigarrow_{10} \qquad \frac{}{[\Delta]_0 + [\cdot]_1 \rightsquigarrow [\Delta]_0, \epsilon} \rightsquigarrow_{01}$$

$$\frac{}{[\Delta_1]_1 + [\Delta_2]_1 \rightsquigarrow [\Delta_1, \Delta_2]_1, \epsilon} \rightsquigarrow_{11}$$

$$\frac{\theta = \mathrm{mgu}(\sigma_1, \sigma_2) \quad [\Delta_1\theta]_{w_1} + [\Delta_2\theta]_{w_2} \rightsquigarrow [\Delta]_w, \xi}{[\Delta_1, A \cdot \sigma_1]_{w_1} + [\Delta_2, A \cdot \sigma_2]_{w_2} \rightsquigarrow [\Delta, A \cdot \sigma_1\theta\xi]_w, \theta\xi} \rightsquigarrow$$

There is considerable non-determinism in the last rule which stems from the assumption that contexts Δ are not ordered. This non-determinism is in addition to the usual overlap between the last two rules.

Theorem 5.18 (Lifted additive contraction)**.**

1. *If* $[\Delta_1]_0 + [\Delta_2]_0 \rightsquigarrow \Delta, \xi$, *then* $\Delta_1\xi = \Delta_2\xi = \Delta$.
2. *If* $[\Delta_1]_1 + [\Delta_2]_0 \rightsquigarrow \Delta, \xi$, *then* $\Delta_1\xi \subseteq \Delta_2\xi = \Delta$.
3. *If* $[\Delta_1]_0 + [\Delta_2]_1 \rightsquigarrow \Delta, \xi$, *then* $\Delta_2\xi \subseteq \Delta_1\xi = \Delta$.
4. *If* $[\Delta_1]_1 + [\Delta_2]_1 \rightsquigarrow \Delta, \xi$, *then* $\Delta_1\xi \subseteq \Delta$ *and* $\Delta_2\xi \subseteq \Delta$.

Proof. Induction on definition 5.17. The only interesting case is for the last rule, "$\rightsquigarrow$", for which we have the following cases.

Case $w_1 = w_2 = 0$, i.e.,

$$\frac{\theta = \mathrm{mgu}(\sigma_1, \sigma_2) \quad [\Delta_1\theta]_0 + [\Delta_2\theta]_0 \rightsquigarrow [\Delta]_0 \cdot \xi}{[\Delta_1, A \cdot \sigma_1]_0 + [\Delta_2, A \cdot \sigma_2]_0 \rightsquigarrow [\Delta, A \cdot \sigma_1\theta\xi]_0 \cdot \theta\xi} \rightsquigarrow$$

$$\Delta_1\theta\xi = \Delta_2\theta\xi = \Delta \qquad \text{i.h.}$$
$$\Delta_1\theta\xi, A \cdot \sigma_1\theta\xi = \Delta_2, A \cdot \sigma_2\theta\xi = \Delta, A \cdot \sigma_1\theta\xi \qquad \theta = \mathrm{mgu}(\sigma_1, \sigma_2).$$

Case $w_1 = w_2 = 1$, i.e.,

$$\frac{\theta = \mathrm{mgu}(\sigma_1, \sigma_2) \quad [\Delta_1\theta]_1 + [\Delta_2\theta]_1 \rightsquigarrow [\Delta]_1 \cdot \xi}{[\Delta_1, A \cdot \sigma_1]_1 + [\Delta_2, A \cdot \sigma_2]_1 \rightsquigarrow [\Delta, A \cdot \sigma_1\theta\xi]_1 \cdot \theta\xi} \rightsquigarrow$$

$$\Delta_1\theta\xi \subseteq \Delta \supseteq \Delta_2\theta\xi \qquad \text{i.h.}$$
$$\Delta_1\theta\xi, A \cdot \sigma_1\theta\xi \subseteq \Delta, A \cdot \sigma_1\theta\xi \supseteq \Delta_2\theta\xi, A \cdot \sigma_2\theta\xi \qquad \theta = \mathrm{mgu}(\sigma_1, \sigma_2).$$

Case $w_1 = 1$ and $w_2 = 0$, i.e.,

$$\frac{\theta = \mathrm{mgu}(\sigma_1, \sigma_2) \quad [\Delta_1\theta]_1 + [\Delta_2\theta]_0 \rightsquigarrow [\Delta]_0 \cdot \xi}{[\Delta_1, A \cdot \sigma_1]_1 + [\Delta_2, A \cdot \sigma_2]_0 \rightsquigarrow [\Delta, A \cdot \sigma_1\theta\xi]_0 \cdot \theta\xi} \rightsquigarrow$$

$$\Delta_1\theta\xi \subseteq \Delta_2\theta\xi = \Delta \qquad \text{i.h.}$$
$$\Delta_1\theta\xi, A \cdot \sigma_1\theta\xi \subseteq \Delta_2\theta\xi, A \cdot \sigma_2\theta\xi = \Delta, A \cdot \sigma_2\theta\xi \qquad \theta = \mathrm{mgu}(\sigma_1, \sigma_2).$$

The case for $w_1 = 0$ and $w_2 = 1$ is similar. □

We now have sufficient machinery to give the rules of the lifted forward calculus. As mentioned earlier, we write the lifted propositions with their corresponding substitutions using a dotted notation, $A \cdot \sigma$, where A is a *free subformula* of the goal sequent and σ is an (idempotent) substitution.

Initial sequents can no longer require the two atomic propositions on the left and right to have syntactically equal free subformulas, but it is sufficient for the two propositions to be unifiable. Thus, the rule for initial sequents is.

$$\frac{\theta = \mathrm{mgu}(\vec{t_1}, \vec{t_2})}{\cdot\,;[p(\vec{t_1}) \cdot \theta]_0 \longrightarrow\!\!\!\!\rightarrow p(\vec{t_2}) \cdot \theta}$$

This rule is not actually sound, because $\vec{t_1}$ and $\vec{t_2}$ are computed from free subformulas, they may inadvertently share variables. We have to force the variables in $\vec{t_1}$ and $\vec{t_2}$ to be distinct by renaming one of them to fresh variables. Thus, the actual "init" rule we use is:

$$\frac{\theta = \mathrm{mgu}(\vec{t_1}[\rho], \vec{t_2})}{\cdot\,;[p(\vec{t_1}) \cdot \rho\theta]_0 \longrightarrow\!\!\!\!\rightarrow p(\vec{t_2}) \cdot \theta}\ \text{init}$$

We use ρ to stand for *renaming substitutions*, that is, substitutions whose range is disjoint from all other variables occurring in the inference rule. In particular, $\mathrm{rng}(\rho)$ is distinct from the variables in $\vec{t_2}$. The variables in a sequent may always be renamed to fresh variables using the following rule.

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma}{\Gamma\rho\,;[\Delta\rho]_w \longrightarrow\!\!\!\!\rightarrow \gamma\rho}\ \text{rename}$$

For binary rules we have to ensure that the operands of the binary connective are unifiable, and if so we assemble a conclusion using the most general unifiers. For example, for $\otimes R$, the rule is as follows:

$$\frac{\Gamma_1\,;[\Delta]_{w_1} \longrightarrow\!\!\!\!\rightarrow A \cdot \sigma_1 \quad \Gamma'\,;[\Delta']_{w_2} \longrightarrow\!\!\!\!\rightarrow B \cdot \sigma_2 \quad \theta = \mathrm{mgu}(\sigma_1, \sigma_2)}{\Gamma_1\theta, \Gamma_2\theta\,;[\Delta_1\theta, \Delta_2\theta]_{w_1 \vee w_2} \longrightarrow\!\!\!\!\rightarrow A \otimes B \cdot \sigma\theta}\ \otimes R$$

In order for this rule to be sound, we require the two premisses $\Gamma_1\,;[\Delta]_{w_1} \longrightarrow\!\!\!\!\rightarrow A \cdot \sigma_1$ and $\Gamma'\,;[\Delta']_{w_2} \longrightarrow\!\!\!\!\rightarrow B \cdot \sigma_2$ so share no free variables. Note that A and B, being free subformulas of the goal sequent, can share variables; however, the result of the substitutions $A[\sigma_1]$ and $B[\sigma_2]$ are forbidden to share any free variables; similarly for the propositions on the left of the sequent arrow. Sequents can be renamed using "rename" as necessary to ensure that the free variables are disjoint.

For additive rules we make use of the lifted additive contraction. For example, for $\&R$ we have the following rule.

$$\frac{\begin{array}{l}\Gamma_1 \mathbin{;} [\Delta_1]_{w_1} \longrightarrow\!\!\!\!\!\rightarrow A \cdot \sigma_1 \\ \Gamma_2 \mathbin{;} [\Delta_2]_{w_2} \longrightarrow\!\!\!\!\!\rightarrow B \cdot \sigma_2 \quad \theta = \mathrm{mgu}(\sigma_1, \sigma_2) \quad [\Delta_1\theta]_{w_1} + [\Delta_2\theta]_{w_2} \rightsquigarrow [\Delta]_w \cdot \xi\end{array}}{\Gamma_1\xi, \Gamma_2\xi \mathbin{;} [\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow A \mathbin{\&} B \cdot \sigma_1\theta\xi}\ \&R$$

The full set of rules is in figures 5.2, 5.3 and 5.4. The notation $\gamma \cup \gamma'$ is understood as follows:

$$\gamma \cup \gamma' = \begin{cases} \cdot & \text{if } \gamma = \gamma' = \cdot \\ \gamma & \text{if } \gamma' = \cdot \\ \gamma' & \text{if } \gamma = \cdot \end{cases}$$

In these rules we use $(\Gamma \mathbin{;} [\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow \gamma)\theta$ as a shorthand for $\Gamma\theta \mathbin{;} [\Delta\theta]_w \longrightarrow\!\!\!\!\!\rightarrow \gamma\theta$. All substitutions are assumed to be idempotent as usual.

In order to show soundness of the lifted forward derivations, we have to first produce ground sequents from lifted sequents, and then argue that *all* ground sequents produced in this fashion are derivable in the ground calculus. In other words, a lifted sequent provides sufficient evidence for all its ground instances.

Definition 5.19 (Grounding substitution). *A substitution σ is a* grounding substitution *for a term (resp. proposition, collection of propositions, and sequent) if for every free term variable x in the term (resp. proposition, collection of propositions, and sequent), the term $x[\sigma]$ contains no term variables.*

Definition 5.20 (Unlifting). *Given a lifted proposition $A \cdot \sigma$, its* unlifted *form, written $\lfloor A \cdot \sigma \rfloor$, is $A[\sigma]$. This definition is extended to contexts pointwise, and for lifted sequents as follows*

$$\lfloor \Gamma \mathbin{;} [\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow \gamma \rfloor = \lfloor \Gamma \rfloor \mathbin{;} [\lfloor \Delta \rfloor]_w \longrightarrow \lfloor \gamma \rfloor$$

Theorem 5.21 (Soundness of the lifted forward calculus). *If $s = \Gamma \mathbin{;} [\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow \gamma$ is derivable and λ is a grounding substitution for s, then $\lfloor \Gamma \mathbin{;} [\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow \gamma \rfloor [\lambda]$ is derivable.*

Judgemental

$$\frac{\theta = \mathrm{mgu}(\vec{t}_1[\rho], \vec{t}_2)}{\cdot\,;[p(\vec{t}_1)\cdot\rho\theta]_0 \longrightarrow\!\!\!\!\rightarrow p(\vec{t}_2)\cdot\theta}\ \text{init} \qquad \frac{\Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma}{(\Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma)\rho}\ \text{rename}$$

$$\frac{\Gamma, A\cdot\sigma_1, A\cdot\sigma_2\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma \quad \theta = \mathrm{mgu}(\sigma_1,\sigma_2)}{(\Gamma, A\cdot\sigma_1\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma)\theta}\ \text{factor}$$

Multiplicative

$$\frac{\Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow A\cdot\sigma_1 \quad \Gamma'\,;[\Delta']_{w'} \longrightarrow\!\!\!\!\rightarrow B\cdot\sigma_2 \quad \theta = \mathrm{mgu}(\sigma_1,\sigma_2)}{(\Gamma,\Gamma'\,;[\Delta,\Delta']_{w\vee w'} \longrightarrow\!\!\!\!\rightarrow A\otimes B\cdot\sigma_1)\theta}\ \otimes R$$

$$\frac{\Gamma\,;[\Delta, A\cdot\sigma_1, B\cdot\sigma_2]_w \longrightarrow\!\!\!\!\rightarrow \gamma \quad \theta = \mathrm{mgu}(\sigma_1,\sigma_2)}{(\Gamma\,;[\Delta, A\otimes B\cdot\sigma_1]_w \longrightarrow\!\!\!\!\rightarrow \gamma)\theta}\ \otimes L \qquad \frac{\Gamma\,;[\Delta, A_i\cdot\sigma]_1 \longrightarrow\!\!\!\!\rightarrow \gamma}{\Gamma\,;[\Delta, A_1\otimes A_2\cdot\sigma]_1 \longrightarrow\!\!\!\!\rightarrow \gamma}\ \otimes L'$$

$$\frac{}{\cdot\,;[\cdot]_0 \longrightarrow\!\!\!\!\rightarrow \mathbf{1}\cdot\epsilon}\ \mathbf{1}R \qquad \frac{\Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma}{\Gamma\,;[\Delta, \mathbf{1}\cdot\epsilon]_w \longrightarrow\!\!\!\!\rightarrow \gamma}\ \mathbf{1}L$$

$$\frac{\Gamma\,;[\Delta, A\cdot\sigma_1]_w \longrightarrow B\cdot\sigma_2 \quad \theta = \mathrm{mgu}(\sigma_1,\sigma_2)}{(\Gamma\,;[\Delta]_w \longrightarrow A\multimap B\cdot\sigma_1)\theta}\ {\multimap}R$$

$$\frac{\Gamma\,;[\Delta, A\cdot\sigma]_1 \longrightarrow\!\!\!\!\rightarrow \cdot}{\Gamma\,;[\Delta]_1 \longrightarrow\!\!\!\!\rightarrow A\multimap B\cdot\sigma}\ {\multimap}R' \qquad \frac{\Gamma\,;[\Delta]_1 \longrightarrow\!\!\!\!\rightarrow B\cdot\sigma}{\Gamma\,;[\Delta]_1 \longrightarrow\!\!\!\!\rightarrow A\multimap B\cdot\sigma}\ {\multimap}R''$$

Figure 5.2: Judgemental and multiplicative rules for the lifted forward calculus

Proof. The proof is by induction on the structure of the derivation $\mathcal{F} :: \Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma$, using theorem 5.18 as needed. In most cases we simply have to note that the mgu is more general than any grounding substitution. The following are few characteristic cases.

Case $\mathcal{F} = \dfrac{\theta = \mathrm{mgu}(\vec{t}_1[\rho], \vec{t}_2)}{\cdot\,;[p(\vec{t}_1)\cdot\rho\theta]_0 \longrightarrow\!\!\!\!\rightarrow p(\vec{t}_2)\cdot\theta}$ init.

For any substitution λ, $\vec{t}_1\rho\theta\lambda = \vec{t}_2\theta\lambda$ as θ is an mgu. Clearly, using the "init" rule, $\cdot\,;[p(\vec{t}_1)[\rho\theta\lambda]]_0 \longrightarrow p(\vec{t}_2)[\theta\lambda]$.

Case $\mathcal{F} = \dfrac{\Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma}{(\Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma)\rho}$ rename.

If λ is a grounding substitution for $(\Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma)\rho$, then $\rho\lambda$ is a grounding substitution for $\Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma$. Then, by the i.h., $\lfloor \Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma \rfloor\, \rho\lambda$.

Additive

$$\frac{\begin{array}{ll}\Gamma_1 \,;\, [\Delta_1]_{w_1} \longrightarrow\!\!\!\!\rightarrow A \cdot \sigma_1 & \theta = \mathrm{mgu}(\sigma_1, \sigma_2) \\ \Gamma_2 \,;\, [\Delta_2]_{w_2} \longrightarrow\!\!\!\!\rightarrow B \cdot \sigma_2 & [\Delta_1\theta]_{w_1} + [\Delta_2\theta]_{w_2} \rightsquigarrow [\Delta]_w \cdot \xi\end{array}}{(\Gamma_1, \Gamma_2 \,;\, [\Delta]_w \longrightarrow\!\!\!\!\rightarrow A \,\&\, B \cdot \sigma_1)\theta\xi} \,\&R$$

$$\frac{\Gamma \,;\, [\Delta, A_i \cdot \sigma]_w \longrightarrow\!\!\!\!\rightarrow \gamma}{\Gamma \,;\, [\Delta, A_1 \,\&\, A_2 \cdot \sigma]_w \longrightarrow\!\!\!\!\rightarrow \gamma} \,\&L_i \qquad \frac{\Gamma \,;\, [\Delta]_w \longrightarrow\!\!\!\!\rightarrow A_i \cdot \sigma}{\Gamma \,;\, [\Delta]_w \longrightarrow\!\!\!\!\rightarrow A_1 \oplus A_2 \cdot \sigma} \,\oplus R_i$$

$$\frac{\begin{array}{lll}\Gamma_1 \,;\, [\Delta_1, A \cdot \sigma_1]_{w_1} \longrightarrow\!\!\!\!\rightarrow \gamma_1 & \tau = \mathrm{mgu}(\gamma_1, \gamma_2) & \\ \Gamma_2 \,;\, [\Delta_2, B \cdot \sigma_2]_{w_2} \longrightarrow\!\!\!\!\rightarrow \gamma_2 & \theta = \mathrm{mgu}(\sigma_1\tau, \sigma_2\tau) & [\Delta_1\theta]_{w_1} + [\Delta_2\theta]_{w_2} \rightsquigarrow [\Delta]_w \cdot \xi\end{array}}{(\Gamma_1, \Gamma_2 \,;\, [\Delta, A \oplus B \cdot \sigma_2]_w \longrightarrow\!\!\!\!\rightarrow \gamma \cup \gamma')\tau\theta\xi} \,\oplus L$$

$$\frac{\begin{array}{lll}\Gamma_1 \,;\, [\Delta_1]_1 \longrightarrow\!\!\!\!\rightarrow \gamma_1 & & \\ \Gamma_2 \,;\, [\Delta_2, B \cdot \sigma]_{w_2} \longrightarrow\!\!\!\!\rightarrow \gamma_2 & \theta = \mathrm{mgu}(\gamma_1, \gamma_2) & [\Delta_1\theta]_1 + [\Delta_2\theta]_{w_2} \rightsquigarrow [\Delta]_w \cdot \xi\end{array}}{(\Gamma_1, \Gamma_2 \,;\, [\Delta, A \oplus B \cdot \sigma]_w \longrightarrow\!\!\!\!\rightarrow \gamma_1 \cup \gamma_2)\theta\xi} \,\oplus L'$$

$$\frac{\begin{array}{lll}\Gamma_1 \,;\, [\Delta_1, A \cdot \sigma]_{w_1} \longrightarrow\!\!\!\!\rightarrow \gamma_1 & & \\ \Gamma_2 \,;\, [\Delta_2]_1 \longrightarrow\!\!\!\!\rightarrow \gamma_2 & \theta = \mathrm{mgu}(\gamma_1, \gamma_2) & [\Delta_1\theta]_{w_1} + [\Delta_2\theta]_1 \rightsquigarrow [\Delta]_w \cdot \xi\end{array}}{(\Gamma_1, \Gamma_2 \,;\, [\Delta, A \oplus B \cdot \sigma]_w \longrightarrow\!\!\!\!\rightarrow \gamma_1 \cup \gamma_2)\theta\xi} \,\oplus L''$$

$$\frac{}{\cdot \,;\, [\cdot]_1 \longrightarrow\!\!\!\!\rightarrow \top \cdot \epsilon} \,\top R \qquad \frac{}{\cdot \,;\, [\mathbf{0} \cdot \epsilon]_1 \longrightarrow\!\!\!\!\rightarrow \cdot} \,\mathbf{0}L$$

Figure 5.3: Additive rules for the lifted forward calculus

Exponential

$$\frac{\Gamma \,;\, [\cdot]_w \longrightarrow\!\!\!\!\rightarrow A \cdot \sigma}{\Gamma \,;\, [\cdot]_0 \longrightarrow\!\!\!\!\rightarrow !A \cdot \sigma} \,!R \qquad \frac{\Gamma, A \cdot \sigma \,;\, [\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma}{\Gamma \,;\, [\Delta, !A \cdot \sigma]_w \longrightarrow\!\!\!\!\rightarrow \gamma} \,!L \qquad \frac{\Gamma \,;\, [\Delta]_0 \longrightarrow\!\!\!\!\rightarrow \gamma}{\Gamma \,;\, [\Delta, !A \cdot \epsilon]_0 \longrightarrow\!\!\!\!\rightarrow \gamma} \,!L'$$

Quantifier

$$\frac{\Gamma \,;\, [\Delta]_w \longrightarrow\!\!\!\!\rightarrow [a/x]A \cdot (\sigma, b/a)}{\Gamma \,;\, [\Delta]_w \longrightarrow\!\!\!\!\rightarrow \forall x.\, A \cdot \sigma} \,\forall R^b \qquad \frac{\Gamma \,;\, [\Delta, A \cdot (\sigma, t/x)]_w \longrightarrow\!\!\!\!\rightarrow \gamma}{\Gamma \,;\, [\Delta, \forall x.\, A \cdot \sigma]_w \longrightarrow\!\!\!\!\rightarrow \gamma} \,\forall L$$

$$\frac{\Gamma \,;\, [\Delta]_w \longrightarrow\!\!\!\!\rightarrow A \cdot (\sigma, t/x)}{\Gamma \,;\, [\Delta]_w \longrightarrow\!\!\!\!\rightarrow \exists x.\, A \cdot \sigma} \,\exists R \qquad \frac{\Gamma \,;\, [\Delta, [a/x]A \cdot (\sigma, b/a)]_w \longrightarrow\!\!\!\!\rightarrow \gamma}{\Gamma \,;\, [\Delta, \exists x.\, A \cdot \sigma]_w \longrightarrow\!\!\!\!\rightarrow \gamma} \,\exists L^b$$

Figure 5.4: Exponential and quantifier rules for the lifted forward calculus

Case $\mathcal{F} = \dfrac{\Gamma, A \cdot \sigma_1, A \cdot \sigma_2\,;[\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow \gamma \quad \theta = \mathrm{mgu}(\sigma_1, \sigma_2)}{(\Gamma, A \cdot \sigma_1\,;[\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow \gamma)\theta}$ factor.

Suppose λ is a grounding substitution for $(\Gamma, A \cdot \sigma_1\,;[\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow \gamma)\theta$. Then $\theta\lambda$ is a grounding substitution for $(\Gamma, A \cdot \sigma_1, A \cdot \sigma_2\,;[\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow \gamma)$. Using the induction hypothesis, therefore, $\lfloor \Gamma, A \cdot \sigma_1, A \cdot \sigma_2\,;[\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow \gamma \rfloor [\theta\lambda]$.

Case $\mathcal{F}$ ends with a logical rule. The arguments for these rules are essentially similar in nature. The following is a characteristic case:

$$\frac{\begin{array}{ll}\mathcal{F}_1 :: \Gamma_1\,;[\Delta_1]_{w_1} \longrightarrow\!\!\!\!\!\rightarrow A \cdot \sigma_1 & \theta = \mathrm{mgu}(\sigma_1, \sigma_2) \\ \mathcal{F}_2 :: \Gamma_2\,;[\Delta_2]_{w_2} \longrightarrow\!\!\!\!\!\rightarrow B \cdot \sigma_2 & [\Delta_1\theta]_{w_1} + [\Delta_2\theta]_{w_2} \rightsquigarrow [\Delta]_w \cdot \xi\end{array}}{(\Gamma_1, \Gamma_2\,;[\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow A \mathbin{\&} B \cdot \sigma_1)\theta\xi}\ \&R$$

Suppose λ is a grounding substitution for $(\Gamma_1, \Gamma_2\,;[\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow A \mathbin{\&} B \cdot \sigma_1)\theta\xi$. Then $\theta\xi\lambda$ is a grounding substitution for both $\Gamma_1\,;[\Delta_1]_{w_1} \longrightarrow\!\!\!\!\!\rightarrow A \cdot \sigma_1$ and $\Gamma_2\,;[\Delta_2]_{w_2} \longrightarrow\!\!\!\!\!\rightarrow B \cdot \sigma_2$ because θ is a unifier. Then we note that $[\lfloor \Delta_1 \rfloor\, \theta\xi\lambda]_{w_1} + [\lfloor \Delta_2 \rfloor\, \theta\xi\lambda]_{w_2} \rightsquigarrow [\lfloor \Delta \rfloor\, \lambda]_w$ by the definition 5.5, and then use $\&R$. □

For the completeness theorem, the lifted calculus will produce sequents whose unlifted form may be more general than that produced in the ground calculus. This sort of proof is usually called a *lifting theorem* in the literature.

Definition 5.22. *Given a context $\Gamma = A_1, \ldots, A_n$ and a sequence of substitutions $\vec{\sigma} = \sigma_1, \ldots, \sigma_n$, write $\Gamma[\vec{\sigma}]$ for the context $A_1[\sigma_1], \ldots, A_n[\sigma_n]$, and $\Gamma \cdot \vec{\sigma}$ for the lifted context $A_1 \cdot \sigma_1, \ldots, A_n \cdot \sigma_n$.*

Theorem 5.23 (Completeness).
Suppose $\Gamma[\vec{\sigma}]\,;[\Delta[\vec{\tau}]]_w \longrightarrow \gamma[\xi]$ is derivable, where each $A \in \Gamma \cup \Delta \cup \gamma$ is a free subformula of the goal sequent. Then there exist substitutions $\vec{\sigma}'$, $\vec{\tau}'$, ξ' λ such that

1. *$\Gamma \cdot \vec{\sigma}'\,;[\Delta \cdot \vec{\tau}']_w \longrightarrow\!\!\!\!\!\rightarrow \gamma \cdot \xi'$; and*
2. *$\vec{\sigma}'\lambda = \vec{\sigma}$, $\vec{\tau}'\lambda = \vec{\tau}$ and $\xi'\lambda = \xi$.*

Proof. Induction on the structure of $\mathcal{F} :: \Gamma[\vec{\sigma}]\,;[\Delta[\vec{\tau}]]_w \longrightarrow \gamma[\xi]$. The essential structure of this proof is fairly standard in the literature; see, for example, [35]. The following are a few representative cases.

Case $\mathcal{F} = \dfrac{\vec{s}[\tau] = \vec{t}[\xi]}{\cdot\,;[p(\vec{s})[\tau]]_0 \longrightarrow p(\vec{t})[\xi]}$ init.

Let ρ be a renaming substitution and write $\theta = \text{mgu}(\vec{s}[\rho], \vec{t})$, which exists because $\rho\tau \uplus \xi$ is a unifier for $\vec{s}$ and $\vec{t}$. By "init", therefore, $\cdot\,;[p(\vec{s}) \cdot \rho\theta]_0 \twoheadrightarrow p(\vec{t}) \cdot \theta$. In this case λ exists by the definition of mgu such that $\theta\lambda = \rho\tau \uplus \xi$.

Case $\mathcal{F} = \dfrac{\Gamma[\vec{\sigma}]\,;[\Delta[\vec{\tau}], A[\sigma_A]]_w \longrightarrow \gamma[\xi]}{\Gamma[\vec{\sigma}], A[\vec{\sigma}_A]\,;[\Delta[\vec{\tau}]]_w \longrightarrow \gamma[\xi]}$ copy.

By the induction hypothesis on the premiss, we pick $\vec{\sigma}'$, σ'_A, $\vec{\tau}'$, ξ' and λ which satisfy (1) and (2).

$$\Gamma \cdot \vec{\sigma}'\,;[\Delta \cdot \vec{\tau}', A \cdot \sigma'_A]_w \twoheadrightarrow \gamma \cdot \xi' \qquad \text{i.h.}$$
$$\Gamma \cdot \vec{\sigma}', A \cdot \sigma'_A\,;[\Delta \cdot \vec{\tau}']_w \twoheadrightarrow \gamma \cdot \xi' \qquad \text{"copy"}$$

Case $\mathcal{F} = \dfrac{\Gamma[\vec{\sigma}], A[\sigma], A'[\sigma']\,;[\Delta[\vec{\tau}]]_w \longrightarrow \gamma[\xi] \quad (A[\sigma] = A'[\sigma'])}{\Gamma[\vec{\sigma}], A[\sigma]\,;[\Delta[\vec{\tau}]]_w \longrightarrow \gamma[\xi]}$ factor.

Because $A[\sigma] = A'[\sigma']$, we can choose to view the above rule application as:

$$\dfrac{\Gamma[\vec{\sigma}], A[\sigma], A[\sigma]\,;[\Delta[\vec{\tau}]]_w \longrightarrow \gamma[\xi]}{\Gamma[\vec{\sigma}], A[\sigma]\,;[\Delta[\vec{\tau}]]_w \longrightarrow \gamma[\xi]} \text{ factor.}$$

By the induction hypothesis on the premiss, we pick $\vec{\sigma}'$, σ_{A1}, σ_{A2}, $\vec{\tau}'$, ξ' and λ which satisfy (1) and (2). In particular, $\sigma_{A1}\lambda = \sigma_{A2}\lambda = \sigma$.

$$\Gamma \cdot \vec{\sigma}', A \cdot \sigma_{A1}, A \cdot \sigma_{A2}\,;[\Delta \cdot \vec{\tau}']_w \twoheadrightarrow \gamma \cdot \xi' \qquad \text{i.h.}$$

Let $\theta = \text{mgu}(\sigma_{A1}, \sigma_{A2})$.

$$\Gamma \cdot \vec{\sigma}'\theta, A \cdot \sigma_{A1}\theta\,;[\Delta \cdot \vec{\tau}'\theta]_w \twoheadrightarrow \gamma \cdot \xi\theta \qquad \text{"factor"}$$

Because λ is also a unifier of σ_{A1} and σ_{A2}, there exists a unique μ such that $\lambda = \theta\mu$. This μ obviously satisfies condition (2).

Case $\mathcal{F}$ ends in a multiplicative rule. Consider, for example,

$$\mathcal{F} = \dfrac{\mathcal{F}_2 :: \Gamma_1[\vec{\sigma}_1]\,;[\Delta_1[\vec{\tau}_1]]_{w_1} \longrightarrow A[\xi|_{\text{vars}(A)}] \quad \Gamma_2[\vec{\sigma}_2]\,;[\Delta_2[\vec{\tau}_2]]_{w_2} \longrightarrow B[\xi|_{\text{vars}(B)}]}{\mathcal{F}_1 :: \Gamma_1[\vec{\sigma}_1], \Gamma_1[\vec{\sigma}_2]\,;[\Delta_1[\vec{\tau}_1], \Delta_2[\vec{\tau}_2]]_{w_1 \vee w_2} \longrightarrow A \otimes B[\xi]} \otimes R.$$

By the induction hypothesis on the premisses $\mathcal{F}_1$ and $\mathcal{F}_2$, we pick the substitutions $\vec{\sigma}'_1$, $\vec{\sigma}'_2$, $\vec{\tau}'_1$, $\vec{\tau}'_2$, ξ_A, ξ_B, λ_1 and λ_2 such that

$$\left.\begin{array}{l}\Gamma_1 \cdot \vec{\sigma}_1' \mathrel{;} [\Delta_1 \cdot \vec{\tau}_1']_{w_1} \longrightarrow\!\!\!\!\to A \cdot \xi_A' \\ \vec{\sigma}_1'\lambda_1 = \vec{\sigma}_1, \vec{\tau}_1'\lambda_1 = \vec{\tau}_1, \text{ and } \xi_A\lambda_1 = \xi|_{\mathrm{vars}(A)}\end{array}\right\} \qquad \text{i.h. on } \mathcal{F}_1$$

$$\left.\begin{array}{l}\Gamma_2 \cdot \vec{\sigma}_2' \mathrel{;} [\Delta_2 \cdot \vec{\tau}_2']_{w_2} \longrightarrow\!\!\!\!\to B \cdot \xi_B' \\ \vec{\sigma}_2'\lambda_2 = \vec{\sigma}_2, \vec{\tau}_2'\lambda_2 = \vec{\tau}_2, \text{ and } \xi_B\lambda_2 = \xi|_{\mathrm{vars}(B)}\end{array}\right\} \qquad \text{i.h. on } \mathcal{F}_2$$

Using "rename", we can guarantee that $\mathrm{dom}(\lambda_1) \cap \mathrm{dom}(\lambda_2) = \emptyset$. Let $\theta = \mathrm{mgu}(\xi_A, \xi_B)$. Then,

$$\Gamma_1 \cdot \vec{\sigma}_1'\theta, \Gamma_2 \cdot \vec{\sigma}_2'\theta \mathrel{;} [\Delta_1 \cdot \vec{\tau}_1'\theta, \Delta_2 \cdot \vec{\tau}_2'\theta]_{w_1 \vee w_2} \longrightarrow\!\!\!\!\to A \otimes B \cdot \xi_A\theta \qquad \otimes R$$

Now, $\lambda = \lambda_1 \uplus \lambda_2$ is also a unifier of ξ_A and ξ_B, so there is a unique μ such that $\lambda = \theta\mu$, and $\xi_A\theta\mu = \xi_A\lambda = \xi_B\lambda = \xi_B\theta\mu$. Thus, this μ satisfies condition (2), for:

$$\Gamma_1 \cdot \vec{\sigma}_1'\theta\mu = \Gamma_1 \cdot \vec{\sigma}_1'\lambda = \Gamma_1 \cdot \vec{\sigma}_1'\lambda_1 = \Gamma_1 \cdot \vec{\sigma}_1$$

and so on.

Case The additive and exponential cases follow similarly. □

5.4 Labelling and sequent representation

By the completeness theorem, we see that the lifted forward calculus produces sequents that are more general than their corresponding sequents in the ground forward calculus. Therefore, if the ground forward calculus produces a derivation that subsumes the given goal sequent, so will the lifted forward calculus. What remains is to detail the algorithm that computes the initial sequents and rule applications based on a given goal sequent.

Recall that we are working with a rectified goal sequent $\Gamma_0 \mathrel{;} \Delta_0 \Longrightarrow C_0$ where the positive $\forall$ and negative $\exists$ bind parameters instead of term variables. For labelling purposes, we simply assign a label to every free subformula of the goal sequent, and for its arguments give it the list of bound parameters or term variables that the subformula was in the scope of. To illustrate, suppose we are labelling the positive proposition $\forall a.\, \exists x.\, (\forall y.\, p(f(a), g(y)) \oplus r) \multimap q(a, x)$. The following is a possible assignment of labels to the subformulas.

$$\begin{array}{rcl} l_0 & \# & \forall a.\, \exists x.\, (\forall y.\, p(f(a), g(y)) \oplus r) \multimap q(a, x) \\ l_1(a) & \# & \exists x.\, (\forall y.\, p(f(a), g(y)) \oplus r) \multimap q(a, x) \end{array}$$

$$\begin{array}{rcl} l_2(a,x) & \# & (\forall y.\, p(f(a), g(y)) \oplus r) \multimap q(a,x) \\ l_3(a,x) & \# & \forall y.\, p(f(a), g(y)) \oplus r \\ l_4(a,x,y) & \# & p(f(a), g(y)) \oplus r \end{array}$$

As usual, we do not label the atomic subformulas. This labelling obviously induces equations among the subformulas; for instance, the above labelling gives:

$$l_0 = \forall a.\, l_1(a) \qquad l_1(a) = \exists x.\, l_2(a,x) \qquad l_2(a,x) = l_3(a,x) \multimap q(a,x) \qquad l_3(a,x) = \forall y.\, l_4(a,x,y)$$

We treat the above induced equations as the definitions of new predicate symbols. For example, we view the above equation for l_3 as $l_3(a,t) = \forall y.\, l_4(a,t,y)$ for every parameter a and term t.

For every one of these induced predicate definitions, we construct a new specialised logical rule. For instance, if we have a definition of a positive label $l(\vec{s}) = l_1(\vec{s}) \otimes l_2(\vec{s})$, then the specialised rule is:

$$\frac{\Gamma_1 \,;\, [\Delta_1]_{w_1} \longrightarrow\!\!\!\!\rightarrow l_1(\vec{s}) \cdot \sigma_1 \quad \Gamma_2 \,;\, [\Delta_2]_{w_2} \longrightarrow\!\!\!\!\rightarrow l_2(\vec{s}) \cdot \sigma_2 \quad \theta = \mathrm{mgu}(\sigma_1, \sigma_2)}{(\Gamma_1, \Gamma_2 \,;\, [\Delta_1, \Delta_2]_{w_1 \vee w_2} \longrightarrow\!\!\!\!\rightarrow l(\vec{s}) \cdot \sigma_1)\theta}\; l_{\otimes R}$$

In the case of quantifiers, we simply perform the relevant eigenvariable check when needed. For instance, if $l(x,a,y) = \forall b.\, l'(x,a,y,b)$, then the specialised rule is

$$\frac{\Gamma \,;\, [\Delta]_w \longrightarrow\!\!\!\!\rightarrow l'(x,a,y,b) \cdot (\sigma, c/b)}{\Gamma \,;\, [\Delta]_w \longrightarrow\!\!\!\!\rightarrow l(x,a,y) \cdot \sigma}\; {l_{\forall R}}^{c}$$

As before, during proof search we limit the applicable logical rules to these specialised rules. The judgemental rules, of course, continue to be generic in nature, as they can apply to all (relevant) labels.

To compute the initial sequents, we consider all positive and negative free atomic predicates, each renamed to fresh variables, and look for pairs of negative and positive atomic predicates that unify. For each pair we produce a new initial sequent according to the "init" rule, and consider it for inclusion in the kept sequents database.

As before with the propositional inverse method, we add a special provision for linear hypotheses that appear more than once. Thus, for each substitution, we also store the multiplicity of that substitution in the sequent; that is, the components of the labelled sequents are of the form $l(\vec{s}) \cdot \sigma^k$ where $l(\vec{s})$ is a free subformula of the goal sequent and k is the multiplicity of the substitution σ. Note that the multiplicity only applies to identical substitutions, not unifiable ones. This is necessary for completeness because, even though the linear context may have two different unifiable propositions, the result of "factoring" them into a common substitution of the requisite multiplicity will produce a sequent that does not subsume the original sequent. Indeed, such an unification may induce spurious equalities in unrelated portions of the sequent. Lifted additive contraction must be extended to handle these multiplicities.

Definition 5.24 (Lifted additive contraction with multiplicities)**.** *The* lifted additive contraction judgement, *written* $[\Delta_1]_{w_1} + [\Delta_2]_{w_2} \rightsquigarrow [\Delta]_w \cdot \xi$*, takes as input the weak-flagged* $[\Delta_1]_{w_1}$ *and* $[\Delta_2]_{w_2}$ *and produces a contracted context* $[\Delta]_w$ *and its corresponding substitution* ξ*. The rules for this judgement are as follows.*

$$\frac{}{[\cdot]_0 + [\cdot]_0 \rightsquigarrow [\cdot]_0 \cdot \epsilon} \rightsquigarrow_{00} \qquad \frac{}{[\cdot]_1 + [\Delta]_0 \rightsquigarrow [\Delta]_0 \cdot \epsilon} \rightsquigarrow_{10} \qquad \frac{}{[\Delta]_0 + [\cdot]_1 \rightsquigarrow [\Delta]_0 \cdot \epsilon} \rightsquigarrow_{01}$$

$$\frac{}{[\Delta_1]_1 + [\Delta_2]_1 \rightsquigarrow [\Delta_1, \Delta_2]_1 \cdot \epsilon} \rightsquigarrow_{11}$$

$$\frac{k = min(m, n) \quad \theta = \mathrm{mgu}(\sigma_1, \sigma_2) \quad [\Delta_1\theta, A \cdot \sigma_1\theta^{m-k}]_{w_1} + [\Delta_2\theta, A \cdot \sigma_2\theta^{n-k}]_{w_2} \rightsquigarrow [\Delta]_w \cdot \xi}{[\Delta_1, A \cdot \sigma_1^m]_{w_1} + [\Delta_2, A \cdot \sigma_2^n]_{w_2} \rightsquigarrow [\Delta, A \cdot \sigma_1\theta\xi^k]_w \cdot \theta\xi} \rightsquigarrow$$

Here, $\Delta, A \cdot \sigma^0$ *is understood as* Δ*.*

In the $\rightsquigarrow$ rule, we contract as many of the propositions as possible. Note that either $m - k$ or $n - k$ in the premiss will be 0. It is easy to see that in the propositional case, i.e., where all substitutions are ϵ, the above definition amounts to computing $\Delta \sqcup \Delta'$ where the multiplicity of any resource is the maximum of its multiplicity in either input context. However, as before, the rules in the first order case are non-deterministic as the $\rightsquigarrow$ and $\rightsquigarrow_{11}$ overlap, and $\rightsquigarrow$ further non-deterministically selects the propositions for contraction.

The implementation of this calculus of course performs contractions eagerly. That is, after every rule application we calculate the possible contractions in the conclusion of the rule. This allows us to limit the contractions to binary rules and consider only the contractions between propositions that originate in different premisses. This is complete because if two hypotheses were to be contractible in the same premiss, then we would already have generated the sequent corresponding to that contraction earlier.

The special case of contracting two weak contexts, i.e., $[\Delta]_1 + [\Delta']_1$, can be greatly improved by first eagerly contracting propositions that have an invertible unifier. This is complete because a weak $\Delta, A \cdot \sigma, A \cdot \sigma\rho$ is a weakening of $\Delta, A \cdot \sigma$.

5.5 Subsumption and indexing

Next we consider two of the most important issues in the first-order case. The first, subsumption, now has to admit instantiation in addition to weakening.

Definition 5.25 (Lifted subsumption)**.** *The* free subsumption *relation* $\prec$ *between lifted forward sequents is the smallest relation satisfying*

$$\left.\begin{array}{r} (\Gamma\,;[\Delta]_0 \longrightarrow\!\!\!\!\!\rightarrow C \cdot \sigma) \prec (\Gamma'\,;[\Delta']_0 \longrightarrow\!\!\!\!\!\rightarrow C \cdot \sigma') \\ (\Gamma\,;[\Delta]_1 \longrightarrow\!\!\!\!\!\rightarrow \gamma) \prec (\Gamma'\,;[\Delta']_w \longrightarrow\!\!\!\!\!\rightarrow \gamma') \end{array}\right\} \begin{array}{l} \text{for some } \theta \text{ such that } \Gamma\theta \subseteq \Gamma', \sigma\theta = \sigma', \\ \Delta\theta \subseteq \Delta' \text{ and } \gamma\theta \subseteq \gamma' \end{array}$$

As mentioned before, the full subsumption test is too expensive always, so in an implementation we optimise for early failure using a sequence of hierarchical tests [111].

Definition 5.26. *The* size *of the context* $\Gamma = A_1 \cdot \sigma_1^{k_1}, \ldots, A_n \cdot \sigma_n^{k_n}$*, written* $\#\Gamma$*, is* $\sum_i k_i$*.*

Definition 5.27 (Hierarchical tests)**.** *A sequent* $s = \Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow \gamma$ does not subsume *another sequent* $s' = \Gamma'\,;[\Delta']_{w'} \longrightarrow\!\!\!\!\!\rightarrow \gamma'$*, written* $s \not\trianglelefteq s'$*, if:*

1. $w = 0$ *and* $w' = 1$*; or*
2. $\#\Delta > \#\Delta'$*, or* $\#\Gamma > \#\Gamma'$*; or*
3. *If the free subformula* A *occurs with multiplicities* $j_1, \ldots, j_m$ *(for its various substitutions) in* Δ *and with multiplicities* $k_1, \ldots, k_n$ *in* Δ' *and* $\sum_i j_i > \sum_i k_i$*; or similarly for* Γ *and* Γ'*; or*
4. *If there is no* θ *such that* $\gamma\theta = \gamma'$*; or*
5. *If for some* $A \cdot \sigma^m \in \Gamma$ *there is no* $A \cdot \tau^n \in \Gamma'$ *such that for no* θ *is* $A[\sigma\theta] = A[\tau]$*; or similarly for* Δ *and* Δ'*; or*
6. *If* $s \not\prec s'$*.*

The following property is rather easy to see. It is the contrapositive of the statement we are interested in, which is that the hierarchical tests must fail if the subsumption has to succeed.

Theorem 5.28 (Soundness of the hierarchical tests). *If $s \ntrianglelefteq s'$, then $s \nprec s'$.*

Proof. Tests 1–3 in defn. 5.27 look at the propositional structure of the contexts; and tests 4 and 5 are both negations of conditions that are required in defn. 5.25. test 6 is, of course, exactly the negation of the definition of lifted subsumption. □

Tammet gives examples of other possible tests in [111], particularly tests that consider the depth of terms and statistics such as the number of constants, but we have found them to be unnecessary in the linear case. The main reason is that linearity considerations—step 2 in defn. 5.27—already account for the overwhelming majority of failed subsumptions. In our experience with many example problems (see chapter 7), it is extraordinarily rare to have situations where the propositional structure would allow for subsumption, but subsumption actually fails because of first-order considerations.[1] (It is of course possible to construct problems where this would be the case.)

Next the issue of indexing into the sequent database. In the search loop, we often have to ask if a new sequent being considered for insertion into the kept sequents database has been generated before (forward subsumption). This question is easy in the propositional case because there is no unification to worry about, but in the first order case we have to of course allow for unification. Thus we have to construct the database in such a fashion that we can efficiently ask for more general forms of a given sequent.

For our implementation of the database, we use a global forest of substitution trees [46, 99]. Substitution trees are what is known as a *perfect filter*: the results of querying a substitution tree gives exactly the answers that satisfy the query. This differs from *imperfect filters* such as discrimination trees [30, 29, 74] or d-trees [46] where there may be extra results that are not relevant to the query. The reason for the imperfection in these filters is that the index forgets important details of the term structure, specifically the identity of the variables, and stores merely the fact that a variable exists in a given position in a term. Substitution trees, on the other hand, store the full term as part of the index, but use substitutions to allow for sophisticated non-local sharing of subterm structure. The price of maintaining full term information is not as high in practice because substitution trees are shallower than discrimination trees.

[1]Note, however, that a more elaborate sequence of hierarchical tests might conceivably give better performance

Each sequent that has to be inserted into the kept sequents database is indexed into the substitution tree corresponding to the label of the principal formula. The index for this sequent is the corresponding substitution of the principal formula. The values corresponding to these indexes are the sequents themselves, which thus form the leaves of the substitution trees in the database.

To check if a given sequent is subsumed, we look up every formula in the sequent in the particular tree in this forest of substitution trees that corresponds to the label of the formula. The particular query we make is: "is the given formula an instance of a formula already stored in the substitution tree?" This is the same as the "instance" query in Graf's description of the substitution tree indexing algorithm [46]. The result of the query is a collection of subsumption candidates. For each candidate, we check if the given sequent is subsumed using the hierarchical tests above.

In the implementation, the subsumption tests are performed iteratively while searching for the subsumption candidates. The order in which these candidates are produced can thus play a crucial role in the efficiency of lookup. Thus we have a range of heuristics to consider when inserting sequents into the database. For instance, our approach of indexing a sequent by the principal formula might not always be the most efficient approach, as that principal formula might be rarely present in future sequents. Tammet has proposed in [111] to index by the *pair* of the principal formula and the "heaviest literal" in the sequent, where the weight of a literal is defined using various factors of its term structure. He presents evidence that this gives a solid benefit to lookup for forward subsumption in the domain of classical hyperresolution. However, his design uses discrimination trees, and it is unclear what the import of his observation is to the linear inverse method.

5.6 Search procedure

The key element in the search procedure is the application of a rule to a given input sequent. We extend the approach in the propositional case in section 4.1.5 by treating all rules as essentially unary rules that produce either a conclusion sequent or a partially instantiated rule. In the first-order case the memoisation of partially applied rules pays off well in preventing repetitions of already successful matches earlier. The price of course is a dynamically growing collection of (partially applied) rules in the rule database, but in practice this price is not too high as there are far fewer rules generated than new sequents. This implementation also has the added benefit of being easily extended to multi-premiss derived rules produced by focusing in chapter 6.

Rule application performs factoring eagerly whenever a new conclusion is derived. In our implementation, we do not in fact have a separate "factor" rule, but rather perform a sequence of factor steps whenever both premisses of a binary rule is matched. As mentioned earlier in sec. 5.4, we do not need to consider factoring two propositions that originate in the same premiss; thus we simply need to consider the additive contraction of the two input unrestricted zones. To illustrate, the binary $\otimes R$ rule is rewritten to:

$$\frac{\begin{array}{l}\Gamma ; [\Delta]_w \longrightarrow\!\!\!\!\!\rightarrow A \cdot \sigma_1 \\ \Gamma' ; [\Delta']_{w'} \longrightarrow\!\!\!\!\!\rightarrow B \cdot \sigma_2 \quad \theta = \mathrm{mgu}(\sigma_1, \sigma_2) \quad [\Gamma\theta]_1 + [\Gamma'\theta]_1 \rightsquigarrow [\Gamma'']_1 \cdot \xi\end{array}}{\Gamma'' ; [\Delta\theta\xi, \Delta'\theta\xi]_{w \vee w'} \longrightarrow\!\!\!\!\!\rightarrow A \otimes B \cdot \sigma_1\theta\xi} \otimes R$$

Note that because the $\rightsquigarrow$ relation is non-deterministic, this rule has several possible conclusions from the same two premisses. All conclusions have to be produced in an implementation of $\rightsquigarrow$. Every produced conclusion will be a consequence of a sequence of "factor" steps from the unfactored sequent (which will also be produced).

As before, our search procedure maintains two databases of sequents as mentioned in defn. 4.17. The inner loop of the search procedure performs the following *lazy activation* step until either the goal sequent is subsumed (in which case the search is successful), or no further rules are applicable to the active sequents, in which case the search saturates. Activation contains a closely related procedure called *percolation* that details the situation where a rule application produced new partially applied rules.

Definition 5.29 (Lazy activation)**.** *To* activate *the sequent s, i.e., to transfer it from the kept sequents database to the active sequents database, the following steps are performed:*

1. *The sequent is renamed and inserted into the active sequents database.*
2. *All available rules are applied to s. If these applications produced new rules R, then percolation (defn. 5.30) is performed on R to obtain the full collection of new partially instantiated rules.*
3. *The new rules are added to the rule database.*

4. *All sequents generated during the above rule applications and percolation, together with all their factors, are tested for subsumption in the global index (forward subsumption). All unsubsumed sequents are added to the kept sequents database.*

Definition 5.30 (Percolation)**.** *To percolate a collection of rules R, the following two steps are performed until there are no new additions to R:*

1. *For every sequent in the active set, every rule in R is applied to it, and*
2. *Any new rules generated are added to R.*

A sequent is added to the kept database if it is not globally subsumed by some sequent derived earlier. In fact, if it is subsumed, then none of its factors need to be computed, as they are merely consequents of a sequent that will not contribute to any further facts. We use the following heuristic for the order of insertion of factors of a given sequent: if s is the result of a sequent of factoring steps from s', then s is checked for subsumption before s'.

5.7 Historical review

The idea of lifting ground derivations to derivations with free variables can be traced back to Robinson's original work on resolution [102]. It is a very general idea that has now become a standard automated reasoning approach; see, for example, its use in logic programming [115]. Ignoring the resource management aspects of this chapter, the essential technical details of lifting a ground forward sequent calculus can be found in the Handbook article on the inverse method [35], which also gives a broader historical perspective.

To the best of our knowledge there has never been (aside from the present work) a consideration of the resource management issues with forward reasoning in first-order linear logic. The problem of automated reasoning for the logic of bunched implications (BI logic) [89] have been attempted by Méry [75] and Donnelly *et al.* [36]. BI logic has many similarities to linear logic, including a common core, but the theorem proving problem is nonetheless harder for BI logic because of its prominent structural rules. Méry's prover uses labelled tableaux in a goal directed fashion, which naturally makes his setting considerably different from forward reasoning. Donnelly *et al.* use the inverse method, but the essential difficulty in their design concerns a particular interaction between weakening and contraction that is germane to bunched implication, but foreign to linear logic. Perhaps because of the difficulty of the problem, their work is in a preliminary form that does not incorporate subsumption or indexing.

Chapter summary *In this chapter the fragment from chapter 3 is extended with the first-order quantifiers. The presence of quantifiers complicates the treatment of additive rules for which we now need sophisticated context comparison processes. This chapter also discusses the updates needed to the inverse method procedure of chapter 4 to handle quantifiers.*

Chapter 6

Focused derivations

The calculi seen so far are *small step* calculi: each logical rule applies to a single logical connective. Of course one can attempt to use such small step calculi for proof search, but in the forward direction this quickly becomes infeasible. The governing engineering problem in the forward direction is that of managing the size of the database of generated sequents. The natural question to ask then is if it is possible to reason in larger steps. Versions of this question have been considered in numerous areas of automated reasoning. In the domain of logic programming, for example, there is the notion of Hereditary Harrop formulas [80] and its generalisation to uniform proof [79] that describes how to treat compound implications as "procedures" in a programming interpretation. Another famous example is *hyperresolution* [103] in the domain of automated theorem proving for classical logic, where the input theory is "cooked" into a clausal form that allows large inferences. Both uniform proofs and hyperresolution are logically motivated foundational approaches, and are therefore fairly generalizable to a wide class of logics.

There have also been investigations into more operational methods of making large inferences. One can, for example, apply chains of unary rules eagerly (in the forward direction). Or one can examine the theory and attempt to extract some extra-logical heuristics for applying rules that will amount to making large inferences. Such approaches are not without merit; however, because these methods are not logically motivated, they tend to be hard to generalize and do not constitute fundamental improvements to search. In this chapter we examine the notion of *focused derivations* that is a logically motivated approach that applies to essentially every non-classical logic. In fact, focusing can be seen as a generalization of both uniform proof search and hyperresolution (see the discussion in sec. 6.4).

This chapter is organised as follows. We start in sec. 6.1 with a formal reconstruction of the focusing (backward) calculus. This calculus will be shown to be complete with respect to the non-focusing calculus by means of a novel cut-elimination proof (thm. 6.7). In sec. 6.2.2 we will then present the forward version of this focusing calculus and sketch the soundness and completeness proofs. In sec. 6.4 we will look at a number of translations of other logics into linear logic and show that on selected fragments the focusing calculus naturally models strategies such as hyperresolution and SLD resolution. In sec. 6.3.1 we will examine the issues with implementing a focusing inverse method prover; the key contribution here will be the construction of derived inference rules in the style of curried functions, together with their use in the lazy activation OTTER loop described in the previous chapter (sec. 5.6).

6.1 Focusing backward sequent calculus

Search using the backward calculus can always apply invertible rules eagerly in any order as there always exists a proof that goes through the premisses of the invertible rule. Andreoli pointed out [7] that a similar and dual feature exists for non-invertible rules also: it is enough for completeness to apply a sequence of non-invertible rules eagerly in one atomic operation, as long as the corresponding connectives are of the same *synchronous* nature. For instance, to infer $p_1 \mathbin{\&} (p_2 \mathbin{\&} p_3)$ on the left, there are three different possible proofs, one for each p_i; these three choices present an essential non-determinism in search. There is never a need to pause with $p_2 \mathbin{\&} p_3$ and consider applying a rule on a different proposition; such a loss of "focus" on $p_2 \mathbin{\&} p_3$ represents an *inessential* non-determinism during proof search. A backward focused proof thus has two phases. In the *active phase* all possible rules are applied in an arbitrary order to asynchronous propositions. When only synchronous propositions remain, one proposition is selected and a *focused phase* for that proposition begins; non-invertible rules are then eagerly (and non-deterministically) applied to decompose that proposition into asynchronous propositions. The proof then again enters the active phase.

In classical linear logic the synchronous or asynchronous nature of a given connective is identical to its polarity; the negative connectives (&, ⊤, ⅋, ⊥, ∀) are asynchronous, and the positive connectives (⊗, **1**, ⊕, **0**, ∃) are synchronous. The nature of intuitionistic connectives, though, must be derived without an appeal to polarity, which is not a primary concept concept in the constructive and judgmental philosophy underlying the logic.[1] We derive this nature by examining the rules and phases of search: an asynchronous connective is one for which decomposition is complete in the active phase; a synchronous connective is one for decomposition is complete in the focused phase. This definition happens to coincide with polarities for classical linear logic, but is decidedly external. The conjunction ∧ from intuitionistic (non-linear) logic, for instance, is nominally of negative polarity but can be seen as both synchronous and asynchronous by our definition; the asynchronous form of the left rule comes from the following left rule:

$$\frac{\Gamma, A, B \Longrightarrow C}{\Gamma, A \wedge B \Longrightarrow C}$$

and the synchronous form of the left rule arises from the pair of left rules:

$$\frac{\Gamma, A \wedge B, A \Longrightarrow C}{\Gamma, A \wedge B \Longrightarrow C} \qquad \frac{\Gamma, A \wedge B, B \Longrightarrow C}{\Gamma, A \wedge B \Longrightarrow C}$$

Either style of the left rule(s) for ∧ by itself would guarantee completeness. (See also sec. 6.4.1.) In classical (non-linear) logic, *every* propositional connective is both synchronous and asynchronous.

As our backward linear sequent calculus is two-sided, we have left- and right- synchronous and asynchronous connectives. For non-atomic propositions a left-synchronous connective is right-asynchronous, and a left-asynchronous connective right-synchronous; this appears to be universal in well-behaved logics. We define the notations in the following table.

symbol	**meaning**
P	left-synchronous (∀, &, ⊤, ⊸)
Q	right-synchronous (∃, ⊗, **1**, ⊕, **0**, !)
L	left-asynchronous (∃, ⊗, **1**, ⊕, **0**, !)
R	right-asynchronous (∀, &, ⊤, ⊸)

[1]Note that polarities may be derived from the synchronous/asynchronous distinction laid out in this section, so it is certainly a definable concept in intuitionistic logics.

The above table does not include the atomic propositions. Andreoli observed in [7] that it is sufficient to assign arbitrarily a synchronous or asynchronous nature to the atoms as long as duality is preserved. However, Andreoli's observation was for classical linear logic where, due to the precise symmetry of connectives, assigning a positive polarity to an atom was equivalent to assigning a negative polarity to its dual, so one would simply obtain the same result in dualised form by flipping the polarity of an atom. However, in our intuitionistic setting since atomic propositions have no deeper propositional structure, we are forced to treat them as synchronous propositions. Andreoli's observation about atoms is not entirely inapplicable to the intuitionistic setting: we can in fact differentiate the atoms by means of a *focusing bias*, which indicates whether the atomic proposition under focus must immediately be derived in an initial sequent. This distinction will become clearer once the details of the calculus are presented.

The backward focusing calculus consists of the following kinds of sequents:

$\Gamma \,;\Delta \gg A$	*right-focal* sequent with A under focus
$\Gamma \,;\Delta \,; A \ll Q$	*left-focal* sequent with A under focus
$\Gamma \,;\Delta \,;\Omega \Longrightarrow C \,; \cdot$	*right-active* sequent
$\Gamma \,;\Delta \,;\Omega \Longrightarrow \cdot \,; Q$	*left-active* sequent

We use γ to represent schematically either right hand form $C \,; \cdot$ or $\cdot \,; Q$. Γ contains the unrestricted resources as usual. Δ contains *only* left-synchronous propositions, i.e., it is of the form $P_1, P_2, \ldots, P_n$. Ω is an ordered context of propositions which may be synchronous or asynchronous, i.e, $A_1 \cdot A_2 \cdots A_n$. We use a centered dot ($\cdot$) instead of a comma to indicate that this context is ordered. In the active sequents, the right propositions in Ω and the proposition C in $C \,; \cdot$ will be called "active".

For active sequents the right active propositions are decomposed until they become right-synchronous, i.e., a sequent of the form $\Gamma \,;\Delta \,;\Omega \Longrightarrow Q \,; \cdot$. The right hand side is then changed into the form $\cdot \,; Q$. Similarly, the propositions in Ω are decomposed except when the proposition is left-synchronous, in which case it is transferred to Δ. The two key judgemental rules that transfer synchronous propositions out of the active zones of the sequents are as follows:

$$\frac{\Gamma \,;\Delta \,;\Omega \Longrightarrow \cdot \,; Q}{\Gamma \,;\Delta \,;\Omega \Longrightarrow Q \,; \cdot}\ \text{ract} \qquad \frac{\Gamma \,;\Delta, P \,;\Omega \cdot \Omega' \Longrightarrow \gamma}{\Gamma \,;\Delta \,;\Omega \cdot P \cdot \Omega' \Longrightarrow \gamma}\ \text{lact}$$

For logical rules, the top level connective in the active proposition is reduced using the corresponding rule in the backward sequent calculus. The following are two characteristic examples:

$$\frac{\Gamma\,;\Delta\,;\Omega \Longrightarrow A\,;\cdot \quad \Gamma\,;\Delta\,;\Omega \Longrightarrow B\,;\cdot}{\Gamma\,;\Delta\,;\Omega \Longrightarrow A \mathbin{\&} B\,;\cdot}\ \&R \qquad \frac{\Gamma\,;\Delta\,;\Omega\cdot A\cdot B\cdot\Omega' \Longrightarrow \gamma}{\Gamma\,;\Delta\,;\Omega\cdot A\otimes B\cdot\Omega' \Longrightarrow \gamma'}\ \otimes L$$

Eventually the active sequent is reduced to the form $\Gamma\,;\Delta\,;\cdot \Longrightarrow \cdot\,;Q$, which we call *neutral sequents*. A focusing phase is launched from such a neutral sequent by selecting a *focile*[2] proposition and giving it the corresponding focus. As mentioned in sec. 6.1, atomic propositions are given focusing biases in our system; if an atom has the wrong bias, it is not considered focile.

Definition 6.1 (Focile propositions)**.**

1. *A proposition is right-focile if it is right-synchronous and not a right-biased atom.*
2. *A proposition is left-focile if it is left-synchronous and not a left-biased atom.*

The following are the rules that give a focile formula in a neutral sequent its corresponding focus.

$$\frac{\Gamma\,;\Delta \gg Q \quad Q \text{ right-focile}}{\Gamma\,;\Delta\,;\cdot \Longrightarrow \cdot\,;Q}\ \text{rfoc} \qquad \frac{\Gamma\,;\Delta\,;P \ll Q \quad P \text{ left-focile}}{\Gamma\,;\Delta,P\,;\cdot \Longrightarrow \cdot\,;Q}\ \text{lfoc}$$

Note that being focile is an internal quality of synchronous propositions. However, there are both synchronous and asynchronous propositions in the unrestricted context Γ. When we are in a neutral sequent, we may copy a proposition out of the unrestricted context and immediately focus on it, regardless of whether it is focile or not.

$$\frac{\Gamma,A\,;\Delta\,;A \ll Q}{\Gamma,A\,;\Delta\,;\cdot \Longrightarrow \cdot\,;Q}\ \text{copy}$$

If this proposition is actually left-asynchronous, then we will immediately remove focus on it and transition to an active phase, as mentioned below. We will use the technical term *blur* to refer to losing focus and transitioning to an active sequent (reading the rules bottom-up).

[2]*Focile* is not standard English and is being used in this thesis to mean "something that can be focused on". We prefer it to "focusable".

Once the focile formula is given focus, is decomposed under focus until it becomes asynchronous or ends in an initial sequent. There are two forms of the initial sequent, corresponding to the two focusing biases.

$$\frac{p \text{ left-biased}}{\Gamma\,;p \gg p}\ \text{rinit} \qquad \frac{p \text{ right-biased}}{\Gamma\,;\cdot\,;p \ll p}\ \text{linit}$$

If the focal proposition becomes atomic, we terminate with one of the two above forms. If the focal proposition is asynchronous, we blur the focus and return to one of the active sequent forms.

$$\frac{\Gamma\,;\Delta\,;\cdot \Longrightarrow R\,;\cdot}{\Gamma\,;\Delta \gg R}\ \text{rb} \qquad \frac{\Gamma\,;\Delta\,;L \Longrightarrow \cdot\,;Q}{\Gamma\,;\Delta\,;L \ll Q}\ \text{lb}$$

Then we are back to an active phase. If the focal proposition is atomic and of the wrong bias, that is the "linit" or "rinit" rules don't apply, then also we blur the focus, but in this case it is not necessary to enter an active phase; instead, we transition directly to the neutral sequent.

$$\frac{\Gamma\,;\Delta\,;\cdot \Longrightarrow \cdot\,;p \quad p \text{ right-biased}}{\Gamma\,;\Delta \gg p}\ \text{rb*} \qquad \frac{\Gamma\,;\Delta,p\,;\cdot \Longrightarrow \cdot\,;Q \quad p \text{ left biased}}{\Gamma\,;\Delta\,;p \ll Q}\ \text{lb*}$$

Decomposing focal propositions uses non-invertible rules for that proposition, and focus is maintained to the operands of the top-level connective of the proposition. The following are a pair of characteristic examples.

$$\frac{\Gamma\,;\Delta_1 \gg A \quad \Gamma\,;\Delta_2 \gg B}{\Gamma\,;\Delta_1,\Delta_2 \gg A \otimes B}\ \otimes R \qquad \frac{\Gamma\,;\Delta\,;A_i \ll Q}{\Gamma\,;\Delta\,;A_1 \,\&\, A_2 \ll Q}\ \& L_i$$

There is only one subtlety with these logical rules, having to do with the exponential connective !. Although it is right synchronous, the $!R$ rule cannot maintain focus on the operand.

$$\frac{\Gamma\,;\cdot\,;\cdot \Longrightarrow A\,;\cdot}{\Gamma\,;\cdot \gg !A}\ !R$$

If we forced the operand to maintain focus, then there would be no focused proof of $\cdot\,;\cdot\,;\cdot \Longrightarrow !(a \oplus b) \multimap !(b \oplus a)\,;\cdot$, for example. To see why, note that the active phase will decompose this sequent to the neutral sequent $a \oplus b\,;\cdot\,;\cdot \Longrightarrow \cdot\,;!(b \oplus a)$. Now we have two choices. If we copy $a \oplus b$ on the left under focus, then we eventually obtain the neutral sequents $a \oplus b \,;\, a \,;\, \cdot \Longrightarrow \cdot \,;\, !(b \oplus a)$ and $a \oplus b \,;\, b \,;\, \cdot \Longrightarrow \cdot \,;\, !(b \oplus a)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{a \oplus b \,;\, a \,;\, \cdot \Longrightarrow \cdot \,;\, !(b \oplus a)}{a \oplus b \,;\, \cdot \,;\, a \Longrightarrow \cdot \,;\, !(b \oplus a)}\,\text{lact} \quad \dfrac{a \oplus b \,;\, b \,;\, \cdot \Longrightarrow \cdot \,;\, !(b \oplus a)}{a \oplus b \,;\, \cdot \,;\, b \Longrightarrow \cdot \,;\, !(b \oplus a)}\,\text{lact}}{a \oplus b \,;\, \cdot \,;\, a \oplus b \Longrightarrow \cdot \,;\, !(b \oplus a)}\,\oplus L}{a \oplus b \,;\, \cdot \,;\, a \oplus b \ll !(b \oplus a)}\,\text{lact}}{a \oplus b \,;\, \cdot \,;\, \cdot \Longrightarrow \cdot \,;\, !(b \oplus a)}\,\text{copy}$$

In either case the $!R$ rule cannot be applied because the linear context is not empty. Thus this choice was wrong and we had to focus on the right, giving $a \oplus b \,;\, \cdot \gg !(b \oplus a)$. If we decompose this under focus to get $a \oplus b \,;\, \cdot \gg b \oplus a$, then the proof cannot proceed because we cannot choose between a and b. However, if we blur the right focus on $b \oplus a$, then we can then focus on the left and get two provable sequents in the premisses of $\oplus L$.

One explanation for this focus-removing behaviour of ! is that there is a hidden transition from $(!\,A)$ *goal* to the categorical judgement A *true* which in turn reduces to A *goal*. We may think of them as two microrules:

$$\dfrac{\Gamma \,;\, \Delta \Longrightarrow A\ \mathit{true}}{\Gamma \,;\, \Delta \Longrightarrow (!\,A)\ \mathit{goal}} \qquad \dfrac{\Gamma \,;\, \cdot \Longrightarrow A\ \mathit{goal}}{\Gamma \,;\, \cdot \Longrightarrow A\ \mathit{true}}$$

The first of these two rules is the internalisation of the categorical judgement and is invertible; the second the second is the definition of the categorical judgement and is non-invertible. The exponential therefore has aspects of both synchronicity and asynchronicity: the overall composition is synchronous, but there is a phase change when applying the rule. Girard has made a similar observation that exponentials are composed of one micro-connective to change polarity, and another to model a given behavior [44, Page 114]; this observation extends to other modal operators, such as why-not (?) of JILL [27] (as in sec. 2.1.3) or the lax modality of CLF [117].

The full set of rules is in fig. 6.1. Soundness of this calculus is rather an obvious property— forget the distinction between Δ and Ω, elide the focus and blur rules, and the original backward calculus appears.

Theorem 6.2 (Soundness)**.**

1. *If* $\Gamma \,;\, \Delta \gg A$ *then* $\Gamma \,;\, \Delta \Longrightarrow A$.
2. *If* $\Gamma \,;\, \Delta \,;\, A \ll Q$ *then* $\Gamma \,;\, \Delta, A \Longrightarrow Q$.

$\boxed{\Gamma\,;\Delta \gg A \qquad \text{right-focal}}$

$$\frac{q \text{ left-biased}}{\Gamma\,;q \gg q}\,\text{linit} \qquad \frac{}{\Gamma\,;\cdot \gg \mathbf{1}}\,\mathbf{1}R \qquad \frac{\Gamma\,;\Delta_1 \gg A \quad \Gamma\,;\Delta_2 \gg B}{\Gamma\,;\Delta_1,\Delta_2 \gg A \otimes B}\,\otimes R$$

$$\frac{\Gamma\,;\Delta \gg A_i}{\Gamma\,;\Delta \gg A_1 \oplus A_2}\,\oplus R_i \qquad \frac{\Gamma\,;\Delta \gg [t/x]A}{\Gamma\,;\Delta \gg \exists x.A}\,\exists R \qquad \frac{\Gamma\,;\cdot\,;\cdot \Longrightarrow A}{\Gamma\,;\cdot \gg\,!A}\,!R$$

$\boxed{\Gamma\,;\Delta\,;A \ll Q \qquad \text{left-focal}}$

$$\frac{p \text{ right-biased}}{\Gamma\,;\cdot\,;p \ll p}\,\text{rinit} \qquad \frac{\Gamma\,;\Delta\,;A_i \ll Q}{\Gamma\,;\Delta\,;A_1 \,\&\, A_2 \ll Q}\,\&L_i$$

$$\frac{\Gamma\,;\Delta_1\,;B \ll Q \quad \Gamma\,;\Delta_2 \gg A}{\Gamma\,;\Delta_1,\Delta_2\,;A \multimap B \ll Q}\,\multimap L \qquad \frac{\Gamma\,;\Delta\,;[t/x]A \ll Q}{\Gamma\,;\Delta\,;\forall x.A \ll Q}\,\forall L$$

$\boxed{\text{focus}}$

$$\frac{\Gamma\,;\Delta\,;P \ll Q \quad P \text{ left-focile}}{\Gamma\,;\Delta,P \Longrightarrow Q}\,\text{lfoc} \qquad \frac{\Gamma\,;\Delta \gg Q \quad Q \text{ right-focile}}{\Gamma\,;\Delta \Longrightarrow Q}\,\text{rfoc} \qquad \frac{\Gamma,A\,;\Delta\,;A \ll Q}{\Gamma,A\,;\Delta \Longrightarrow Q}\,\text{copy}$$

$\boxed{\Gamma\,;\Delta\,;\Omega \Longrightarrow R\,;\cdot \qquad \text{right-active}}$

$$\frac{\Gamma\,;\Delta\,;\Omega \Longrightarrow A\,;\cdot \quad \Gamma\,;\Delta\,;\Omega \Longrightarrow B\,;\cdot}{\Gamma\,;\Delta\,;\Omega \Longrightarrow A \,\&\, B\,;\cdot}\,\&R \qquad \frac{}{\Gamma\,;\Delta\,;\Omega \Longrightarrow \top\,;\cdot}\,\top R$$

$$\frac{\Gamma\,;\Delta\,;\Omega \cdot A \Longrightarrow B\,;\cdot}{\Gamma\,;\Delta\,;\Omega \Longrightarrow A \multimap B\,;\cdot}\,\multimap R \qquad \frac{\Gamma\,;\Delta\,;\Omega \Longrightarrow [a/x]A\,;\cdot}{\Gamma\,;\Delta\,;\Omega \Longrightarrow \forall x.\,A\,;\cdot}\,\forall R^a \qquad \frac{\Gamma\,;\Delta\,;\Omega \Longrightarrow \cdot\,;Q}{\Gamma\,;\Delta\,;\Omega \Longrightarrow Q\,;\cdot}\,\text{ract}$$

$\boxed{\Gamma\,;\Delta\,;\Omega \cdot L \cdot \Omega' \Longrightarrow \gamma \qquad \text{left-active}}$

$$\frac{\Gamma\,;\Delta\,;\Omega \cdot A \cdot B \cdot \Omega' \Longrightarrow \gamma}{\Gamma\,;\Delta\,;\Omega \cdot A \otimes B \cdot \Omega' \Longrightarrow \gamma}\,\otimes L \qquad \frac{\Gamma\,;\Delta\,;\Omega \cdot \Omega' \Longrightarrow \gamma}{\Gamma\,;\Delta\,;\Omega \cdot \mathbf{1} \cdot \Omega' \Longrightarrow \gamma}\,\mathbf{1}L$$

$$\frac{\Gamma\,;\Delta\,;\Omega \cdot A \cdot \Omega' \Longrightarrow Q \quad \Gamma\,;\Delta\,;\Omega \cdot B \cdot \Omega' \Longrightarrow \gamma}{\Gamma\,;\Delta\,;\Omega \cdot A \oplus B \cdot \Omega' \Longrightarrow \gamma}\,\oplus L \qquad \frac{}{\Gamma\,;\Delta\,;\Omega \cdot \mathbf{0} \cdot \Omega' \Longrightarrow \gamma}\,\mathbf{0}L$$

$$\frac{\Gamma\,;\Delta\,;\Omega \cdot [a/x]A \cdot \Omega' \Longrightarrow \gamma}{\Gamma\,;\Delta\,;\Omega \cdot \exists x.A \cdot \Omega' \Longrightarrow \gamma}\,\exists L^a \qquad \frac{\Gamma,A\,;\Delta\,;\Omega \cdot \Omega' \Longrightarrow \gamma}{\Gamma\,;\Delta\,;\Omega \cdot\, !A \cdot \Omega' \Longrightarrow \gamma}\,!L \qquad \frac{\Gamma\,;\Delta,P\,;\Omega \cdot \Omega' \Longrightarrow \gamma}{\Gamma\,;\Delta\,;\Omega \cdot P \cdot \Omega' \Longrightarrow \gamma}\,\text{lact}$$

$\boxed{\text{blur}}$

$$\frac{\Gamma\,;\Delta\,;L \Longrightarrow \cdot\,;Q}{\Gamma\,;\Delta\,;L \ll Q}\,\text{lb} \qquad \frac{\Gamma\,;\Delta,q\,;\cdot \Longrightarrow \cdot\,;Q \quad q \text{ left-biased}}{\Gamma\,;\Delta\,;q \ll Q}\,\text{lb*}$$

$$\frac{\Gamma\,;\Delta\,;\cdot \Longrightarrow R\,;\cdot}{\Gamma\,;\Delta \gg R}\,\text{rb} \qquad \frac{\Gamma\,;\Delta\,;\cdot \Longrightarrow \cdot\,;p \quad p \text{ right-biased}}{\Gamma\,;\Delta \gg p}\,\text{rb*}$$

Figure 6.1: Backward linear focusing calculus

3. *If* $\Gamma ; \Delta ; \Omega \Longrightarrow C ; \cdot$ *then* $\Gamma ; \Delta, \Omega \Longrightarrow C$.
4. *If* $\Gamma ; \Delta ; \Omega \Longrightarrow \cdot ; Q$ *then* $\Gamma ; \Delta, \Omega \Longrightarrow Q$.

Proof. By structural induction on the given focused derivation. Note that all the logical rules neatly fall into one of the above cases. To illustrate, consider the rule $\otimes R$, i.e, the derivation that ends with the following rule:

$$\frac{\Gamma ; \Delta_1 \gg A \quad \Gamma ; \Delta_2 \gg B}{\Gamma ; \Delta_1, \Delta_2 \gg A \otimes B}$$

$\Gamma ; \Delta_1 \Longrightarrow A$ and $\Gamma ; \Delta_2 \Longrightarrow B$ — i.h.
$\Gamma ; \Delta_1, \Delta_2 \Longrightarrow A \otimes B$ — $\otimes R$.

For phase transition rules (i.e., lb, lb*, rb, rb*, lact, ract, lfoc, and rfoc), the premiss and the conclusion of the rule both denote the same sequent in the non-focusing calculus. □

Theorem 6.3 (Structural properties)**.**

1. *Weakening:*
   (a) *If* $\Gamma ; \Delta \gg A$ *then* $\Gamma, \Gamma' ; \Delta \gg A$.
   (b) *If* $\Gamma ; \Delta ; A \ll Q$ *then* $\Gamma, \Gamma' ; \Delta ; A \ll Q$.
   (c) *If* $\Gamma ; \Delta ; \Omega \Longrightarrow \gamma$ *then* $\Gamma, \Gamma' ; \Delta ; \Omega \Longrightarrow \gamma$.
2. *Contraction:*
   (a) *If* $\Gamma, A, A ; \Delta \gg C$ *then* $\Gamma, A ; \Delta \gg C$.
   (b) *If* $\Gamma, A, A ; \Delta ; B \ll Q$ *then* $\Gamma, A ; \Delta ; B \ll Q$.
   (c) *If* $\Gamma, A, A ; \Delta ; \Omega \Longrightarrow \gamma$ *then* $\Gamma, A ; \Delta ; \Omega \Longrightarrow \gamma$.

Proof sketch. By straightforward structural induction on the given derivations. As before with theorem 2.17, we note that the unrestricted context of any given sequent persist all the way up to the axiomatic cases of the proof branch with that sequent as the conclusion, wherein these structural statements are trivially true. □

We show the completeness of the focusing calculus by interpreting every backward sequent as an active sequent in the focusing calculus, then showing that the backward rules are admissible in the focusing calculus. This proof relies on admissibility of cut in the focusing calculus. Because a non-atomic right-synchronous proposition is left-asynchronous, all principal cuts will be between a focal sequent and an active sequent. For example, for the principal cut for ⊗, we have to consider the following pair of derivations.

$$\dfrac{\Gamma\,;\Delta_1 \gg A \quad \Gamma\,;\Delta_2 \gg B}{\Gamma\,;\Delta_1,\Delta_2 \gg A \otimes B} \qquad \dfrac{\Gamma\,;\Delta'\,;\Omega \cdot A \cdot B \cdot \Omega' \Longrightarrow \gamma}{\Gamma\,;\Delta'\,;\Omega \cdot A \otimes B \cdot \Omega' \Longrightarrow \gamma}$$

The cut is permuted to the component derivations which also maintain this form of cut. A similar situation occurs in the dual case. The result of these cuts will be active because the proposition under focus is cut.

We also have to include a few more general cuts in order for the commutative cases in the cut theorem to work. Primarily, we require cuts between two active sequents, the result of which will be another active sequent. In the proof we also need to consider two special cases where the cut formula is in a focal sequent but not itself under focus. For the induction in the cut theorem to work, these specific cases will have to redo the focusing steps for the proposition under focus, as explained in the details of the proof.

The proof of cut-elimination requires one key lemma: that permuting the ordered context does not affect provability. This lemma thus allows cutting formulas from anywhere inside the ordered context, and also to re-order the context when needed.

Lemma 6.4 (Permutation)**.**
If $\Gamma\,;\Delta\,;\Omega \Longrightarrow \gamma$*, then* $\Gamma\,;\Delta\,;\Omega' \Longrightarrow \gamma$ *for any permutation* Ω' *of* Ω*.* □

Proof. By structural induction on the derivation $\mathcal{D} :: \Gamma\,;\Delta\,;\Omega \Longrightarrow \gamma$. The following is a representative case for ⊗*L*, where $\Omega = \Omega_1 \cdot A \otimes B \cdot \Omega_2$ and the last rule in $\mathcal{D}$ was:

$$\dfrac{\Gamma\,;\Delta\,;\Omega_1 \cdot A \cdot B \cdot \Omega_2 \Longrightarrow \gamma}{\Gamma\,;\Delta\,;\Omega_1 \cdot A \otimes B \cdot \Omega_2 \Longrightarrow \gamma}\,\otimes L$$

Let a permutation Ω' of $\Omega_1 \cdot A \otimes B \cdot \Omega_2$ be given. It has the form $\Omega_1' \cdot A \otimes B \cdot \Omega_2'$ where $\Omega_1' \cdot \Omega_2'$ is a permutation of $\Omega_1 \cdot \Omega_2$, i.e., $\Omega_1' \cdot A \cdot B \cdot \Omega_2'$ is a permutation of $\Omega_1 \cdot A \cdot B \cdot \Omega_2$. Therefore, by the induction hypothesis, $\Gamma\,;\Delta\,;\Omega_1' \cdot A \cdot B \cdot \Omega_2' \Longrightarrow \gamma$. Then use ⊗*L*. □

One consequence of this lemma is that the order of the propositions in the active contexts does not matter. Therefore, we can always find a proof where the decompositions in the active phase fix a canonical order of decomposition. In our implementation, we first decompose the active propositions on the right, and then the left active propositions in the order of right to left. The actual active rules we thus implement in sec. 6.2 interpret the active rules as if they were the following (representative cases):

$$\frac{\Gamma\,;\Delta\,;\Omega \Longrightarrow A\,;\cdot \quad \Gamma\,;\Delta\,;\Omega \Longrightarrow B\,;\cdot}{\Gamma\,;\Delta\,;\Omega \Longrightarrow A \mathbin{\&} B\,;\cdot} \qquad \frac{\Gamma\,;\Delta\,;\Omega\cdot A\cdot B \Longrightarrow \cdot\,;Q}{\Gamma\,;\Delta\,;\Omega\cdot A\otimes B \Longrightarrow \cdot\,;Q}$$

Any other ordering would also work, in principle. We prefer this ordering because it is slightly more systematic; for example, for iterated implications on the right, this ordering would first transfer the antecedents of the implication to the left active context before examining the antecedents in order.

Definition 6.5 (Similar derivations). *Two derivations $\mathcal{D}_1$ and $\mathcal{D}_2$ of $\Gamma\,;\Delta\,;\Omega \Longrightarrow \gamma$, written $\mathcal{D}_1 \approx \mathcal{D}_2$, are said to be* similar *if they differ only in the order in which active rules are applied to elements of Ω and γ.*

Essentially, two derivations are similar if the only differences are in the inessential non-deterministic choices in the active phase. This definition comes with a fact for which we omit the easy proof.

Fact 6.6. *If $\mathcal{D} :: \Gamma\,;\Delta\,;\Omega \Longrightarrow \gamma$ and $\mathcal{D}' \approx \mathcal{D}$, then $\mathcal{D}' :: \Gamma\,;\Delta\,;\Omega \Longrightarrow \gamma$.* □

For the cut theorem, similar derivations are considered to be equal for the purposes of the lexicographic order.

Theorem 6.7 (cut). *If*

1. *$\Gamma\,;\Delta \gg A$ and:*
   (a) *$\Gamma\,;\Delta'\,;\Omega\cdot A\cdot\Omega' \Longrightarrow \gamma$ then $\Gamma\,;\Delta,\Delta'\,;\Omega\cdot\Omega' \Longrightarrow \gamma$.*
   (b) *$\Gamma\,;\Delta',A\,;\Omega \Longrightarrow \gamma$ then $\Gamma\,;\Delta,\Delta'\,;\Omega \Longrightarrow \gamma$.*
2. *$\Gamma\,;\cdot \gg A$ and $\Gamma,A\,;\Delta\,;\Omega \Longrightarrow \gamma$ then $\Gamma\,;\Delta\,;\Omega \Longrightarrow \gamma$.*
3. *$\Gamma\,;\Delta\,;\Omega \Longrightarrow A\,;\cdot$ or $\Gamma\,;\Delta\,;\Omega \Longrightarrow \cdot\,;A$ and:*
   (a) *$\Gamma\,;\Delta'\,;A \ll Q$ then $\Gamma\,;\Delta,\Delta'\,;\Omega \Longrightarrow \cdot\,;Q$.*
   (b) *$\Gamma\,;\Delta'\,;\Omega'\cdot A\cdot\Omega'' \Longrightarrow \gamma$ then $\Gamma\,;\Delta,\Delta'\,;\Omega\cdot\Omega'\cdot\Omega'' \Longrightarrow \gamma$.*
   (c) *$\Gamma\,;\Delta',A\,;\Omega' \Longrightarrow \gamma$ then $\Gamma\,;\Delta,\Delta'\,;\Omega\cdot\Omega' \Longrightarrow \gamma$.*
4. *$\Gamma\,;\cdot\,;\cdot \Longrightarrow A\,;\cdot$ or $\Gamma\,;\cdot\,;\cdot \Longrightarrow \cdot\,;A$ and:*
   (a) *$\Gamma,A\,;\Delta\,;\Omega \Longrightarrow \gamma$ then $\Gamma\,;\Delta\,;\Omega \Longrightarrow \gamma$.*

(b) $\Gamma, A \,;\, \Delta \gg B$ *then* $\Gamma \,;\, \Delta \gg B$.

5. $\Gamma \,;\, \Delta \,;\, B \ll A$ *and:*

   (a) $\Gamma \,;\, \Delta' \,;\, A \Longrightarrow \cdot \,;\, Q$ *then* $\Gamma \,;\, \Delta, \Delta' \,;\, B \ll Q$.

   (b) $\Gamma \,;\, \Delta', A \,;\, \cdot \Longrightarrow \cdot \,;\, Q$ *then* $\Gamma \,;\, \Delta, \Delta' \,;\, B \ll Q$.

Proof. By lexicographic induction on the given derivations. The argument is lengthy rather than complex, and is an adaptation of the proof of theorem 2.21. Name the three derivations in each case $\mathcal{D}$, $\mathcal{E}$ and $\mathcal{F}$ respectively. The lexicographic order prescribed in the proof of theorem 2.21 is extended in the obvious fashion to the focusing calculus, with one further condition that cuts of type 5 may be used in the inductive arguments of all other types of cuts.

A sequent is smaller than another if it has fewer elements in the zones of the context; the order of Ω is irrelevant in comparing sizes of sequents. We can successfully do this because lem. 6.4 guarantees that the precise order of Ω is irrelevant. For the purposes of this proof, derivations of sequents that only differ in the order of the unrestricted contexts are taken to be equal.

Initial cuts. In this cut one of the derivations is initial. For example:

$$\mathcal{D} = \frac{p(\vec{t}) \text{ right-biased}}{\Gamma \,;\, \cdot \,;\, p(\vec{t}) \ll p(\vec{t})} \text{ rinit} \qquad \mathcal{E} :: \Gamma \,;\, \Delta \,;\, \Omega \Longrightarrow p(\vec{t}) \,;\, \cdot$$

Here $\mathcal{F} = \mathcal{E}$. The companion case for left-biased atoms is similar.

Principal cuts. A principal formula is introduced in both $\mathcal{D}$ and $\mathcal{E}$.

Case of $\otimes$:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma \,;\, \Delta_1 \gg A \quad \mathcal{D}_2 :: \Gamma \,;\, \Delta_2 \gg B}{\Gamma \,;\, \Delta_1, \Delta_2 \gg A \otimes B} \otimes R \qquad \mathcal{E} = \frac{\mathcal{E}' :: \Gamma \,;\, \Delta \,;\, \Omega \cdot A \cdot B \cdot \Omega' \Longrightarrow \gamma}{\Gamma \,;\, \Delta \,;\, \Omega \cdot A \otimes B \cdot \Omega' \Longrightarrow \gamma} \otimes L$$

$$\begin{array}{ll} \Gamma \,;\, \Delta_2, \Delta \,;\, \Omega \cdot A \cdot \Omega' \Longrightarrow \cdot \,;\, Q & \text{cut on } \mathcal{D}_2 \text{ and } \mathcal{E}' \\ \Gamma \,;\, \Delta_1, \Delta_2, \Delta \,;\, \Omega \cdot \Omega' \Longrightarrow \cdot \,;\, Q & \text{cut on } \mathcal{D}_1 \text{ and above} \end{array}$$

Case of $\mathbf{1}$:

$$\mathcal{D} = \frac{}{\Gamma \,;\, \cdot \,;\, \cdot \gg \mathbf{1}} \mathbf{1}R \qquad \mathcal{E} = \frac{\mathcal{E}' :: \Gamma \,;\, \Delta \,;\, \Omega \cdot \Omega' \Longrightarrow \gamma}{\Gamma \,;\, \Delta \,;\, \Omega \cdot \mathbf{1} \cdot \Omega' \Longrightarrow \gamma} \mathbf{1}L$$

Here $\mathcal{F} = \mathcal{E}'$.

Case of $\oplus$:

$$\mathcal{D} = \frac{\mathcal{D}' :: \Gamma \,;\Delta \gg A}{\Gamma \,;\Delta \gg A \oplus B}\ \oplus R_1 \qquad \mathcal{E} = \frac{\mathcal{E}_1 :: \Gamma \,;\Delta' \,;\Omega \cdot A \cdot \Omega' \Longrightarrow \gamma \quad \mathcal{E}_2 :: \Gamma \,;\Delta' \,;\Omega \cdot B \cdot \Omega' \Longrightarrow \gamma}{\Gamma \,;\Delta' \,;\Omega \cdot A \oplus B \cdot \Omega' \Longrightarrow \gamma}\ \oplus L$$

$\Gamma \,;\Delta, \Delta' \,;\Omega \Longrightarrow \cdot\,;Q$ cut on $\mathcal{D}'$ and $\mathcal{E}_1$

The case of $\oplus R_2$ is similar.

Case of $\mathbf{0}$: there are no principal cuts for $\mathbf{0}$.

Case of $!$:

$$\mathcal{D} = \frac{\mathcal{D}' :: \Gamma \,;\cdot\,;\cdot \Longrightarrow A\,;\cdot}{\Gamma \,;\cdot \gg\, !A}\ !R \qquad \mathcal{E} = \frac{\mathcal{E}' :: \Gamma, A \,;\Delta \,;\Omega \cdot \Omega' \Longrightarrow Q}{\Gamma \,;\Delta \,;\Omega \cdot !A \cdot \Omega' \Longrightarrow Q}\ !L$$

$\Gamma \,;\Delta \,;\Omega \Longrightarrow Q$ cut on $\mathcal{D}'$ and $\mathcal{E}'$

Case of $\exists$:

$$\mathcal{D} = \frac{\mathcal{D}' :: \Gamma \,;\Delta \gg [t/x]A}{\Gamma \,;\Delta \gg \exists x.A}\ \exists R \qquad \mathcal{E} = \frac{\mathcal{E}' :: \Gamma \,;\Delta' \,;\Omega \cdot [a/x]A \cdot \Omega' \Longrightarrow \gamma}{\Gamma \,;\Delta' \,;\Omega \cdot \exists x.A \cdot \Omega' \Longrightarrow \gamma}\ \exists L^a$$

$\Gamma \,;\Delta, \Delta' \,;\Omega \cdot \Omega' \Longrightarrow \gamma$ cut on $\mathcal{D}'$ and $[t/a]\mathcal{E}'$

Case of $\multimap$:

$$\mathcal{D} = \frac{\mathcal{D}' :: \Gamma \,;\Delta \,;\Omega \cdot A \Longrightarrow B\,;\cdot}{\Gamma \,;\Delta \,;\Omega \Longrightarrow A \multimap B\,;\cdot}\ \multimap R \qquad \mathcal{E} = \frac{\mathcal{E}_1 :: \Gamma \,;\Delta_1 \,;B \ll Q \quad \mathcal{E}_2 :: \Gamma \,;\Delta_2 \gg A}{\Gamma \,;\Delta_1, \Delta_2 \,;A \multimap B \ll Q}\ \multimap L$$

$\Gamma \,;\Delta_2, \Delta \,;\Omega \Longrightarrow B\,;\cdot$ cut on $\mathcal{E}_2$ and $\mathcal{D}'$

$\Gamma \,;\Delta_1, \Delta_2, \Delta \,;\Omega \Longrightarrow \cdot\,;Q$ cut on above and $\mathcal{E}_1$

Case of $\&$:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma \,;\Delta' \,;\Omega \Longrightarrow A\,;\cdot \quad \mathcal{D}_2 :: \Gamma \,;\Delta' \,;\Omega \Longrightarrow B\,;\cdot}{\Gamma \,;\Delta' \,;\Omega' \Longrightarrow A \,\&\, B\,;\cdot}\ \&R \qquad \mathcal{E} = \frac{\mathcal{E}' :: \Gamma \,;\Delta \,;A \ll Q}{\Gamma \,;\Delta \,;A \,\&\, B \ll Q}\ \&L_1$$

$\Gamma \,;\Delta, \Delta' \,;\Omega \Longrightarrow \cdot\,;Q$ cut on $\mathcal{D}'$ and $\mathcal{E}_1$

Case of $\top$: there are no principal cuts for $\top$.

Case of $\forall$:

$$\mathcal{D} = \frac{\mathcal{D}' :: \Gamma \,;\Delta \,;\Omega \Longrightarrow [a/x]A\,;\cdot}{\Gamma \,;\Delta \,;\Omega \Longrightarrow \forall x.A\,;\cdot}\ \forall R^a \qquad \mathcal{E} = \frac{\mathcal{E}' :: \Gamma \,;\Delta' \,;[t/x]A \ll Q}{\Gamma \,;\Delta' \,;\forall x.A \ll Q}\ \forall L$$

$\Gamma\,;\Delta,\Delta'\,;\Omega \Longrightarrow \cdot\,;Q$ cut on $[t/a]\mathcal{D}'$ and $\mathcal{E}'$.

Focus cuts. Where the last rule in $\mathcal{D}$ gives focus to the cut-formula.

Case. $\mathcal{D} = \dfrac{\mathcal{D}' :: \Gamma\,;\Delta\,;P \ll Q}{\Gamma\,;\Delta,P\,;\cdot \Longrightarrow \cdot\,;Q}$ and $\mathcal{E} :: \Gamma\,;\Delta'\,;\Omega \Longrightarrow P\,;\cdot$.

$\Gamma\,;\Delta,\Delta'\,;\Omega \Longrightarrow Q$ cut on $\mathcal{D}'$ and $\mathcal{E}$

The case for $\mathcal{E} :: \Gamma\,;\Delta'\,;\Omega \Longrightarrow \cdot\,;P$ is similar.

Case. $\mathcal{D} = \dfrac{\mathcal{D}' :: \Gamma,A\,;\Delta\,;A \ll Q}{\Gamma,A\,;\Delta\,;\cdot \Longrightarrow \cdot\,;Q}$ and $\mathcal{E} :: \Gamma\,;\Delta'\,;\Omega \Longrightarrow A\,;\cdot$.

$\Gamma,A\,;\Delta'\,;\Omega \Longrightarrow A\,;\cdot$ weakening on $\mathcal{E}$ (thm. 6.3)

$\Gamma,A\,;\Delta,\Delta'\,;\Omega \Longrightarrow \cdot\,;Q$ cut on $\mathcal{D}'$ and above

The case for $\mathcal{E} :: \Gamma\,;\Delta'\,;\Omega \Longrightarrow \cdot\,;A$ is similar.

Case. $\mathcal{D} = \dfrac{\mathcal{D}' :: \Gamma\,;\Delta \gg Q}{\Gamma\,;\Delta\,;\cdot \Longrightarrow \cdot\,;Q}$ and $\mathcal{E} :: \Gamma\,;\Delta'\,;\Omega\cdot Q\cdot\Omega' \Longrightarrow \gamma$.

$\Gamma\,;\Delta,\Delta'\,;\Omega\cdot\Omega' \Longrightarrow \gamma$ cut on $\mathcal{D}'$ and $\mathcal{E}$ ($\mathcal{D}'$ smaller)

Case. $\mathcal{D} = \dfrac{\mathcal{D}' :: \Gamma\,;\cdot \gg Q}{\Gamma\,;\cdot\,;\cdot \Longrightarrow \cdot\,;Q}$ and $\mathcal{E} :: \Gamma,Q\,;\Delta\,;\Omega \Longrightarrow \gamma$.

$\Gamma\,;\Delta\,;\Omega \Longrightarrow \gamma$ cut on $\mathcal{D}'$ and $\mathcal{E}$ ($\mathcal{D}'$ smaller)

Blur cuts. Where the last rule in $\mathcal{E}$ blurs focus from the cut formula.

Case. $\mathcal{E} = \dfrac{\mathcal{E}' :: \Gamma\,;\Delta\,;L \Longrightarrow \cdot\,;Q}{\Gamma\,;\Delta\,;L \ll Q}$ and $\mathcal{D} :: \Gamma\,;\Delta'\,;\Omega' \Longrightarrow L\,;\cdot$.

$\Gamma\,;\Delta,\Delta'\,;\Omega \Longrightarrow \cdot\,;Q$ cut on $\mathcal{D}$ and $\mathcal{E}'$ ($\mathcal{E}'$ smaller)

The case of $\mathcal{D} :: \Gamma\,;\Delta'\,;\Omega' \Longrightarrow \cdot\,;L$ is similar.

Case. $\mathcal{D} = \dfrac{\mathcal{D}' :: \Gamma\,;\Delta\,;\cdot \Longrightarrow R\,;\cdot}{\Gamma\,;\Delta \gg R}$.

Subcase. $\mathcal{E} :: \Gamma\,;\Delta'\,;\Omega\cdot R\cdot\Omega' \Longrightarrow \gamma$.

$\Gamma\,;\Delta,\Delta'\,;\Omega\cdot\Omega' \Longrightarrow \gamma$ cut on $\mathcal{D}'$ and $\mathcal{E}$

Subcase. $\mathcal{E} :: \Gamma\,;\Delta',R\,;\Omega \Longrightarrow \gamma$.

$\Gamma\,;\Delta,\Delta'\,;\Omega \Longrightarrow \gamma$ cut on $\mathcal{D}'$ and $\mathcal{E}$

Case. $\mathcal{D} = \dfrac{\mathcal{D}' :: \Gamma\,;\cdot\,;\cdot \Longrightarrow \cdot\,;R}{\Gamma\,;\cdot \gg R}$ and $\mathcal{E} :: \Gamma,R\,;\Delta\,;\Omega \Longrightarrow \gamma$.

$\Gamma\,;\Delta\,;\Omega \Longrightarrow \gamma$ cut on $\mathcal{D}'$ and $\mathcal{E}$

Case. $\mathcal{D} = \dfrac{\mathcal{D}' :: \Gamma\,;\Delta\,;\cdot \Longrightarrow \cdot\,;p \quad p \text{ right-biased}}{\Gamma\,;\Delta \gg p}$.

Subcase. $\mathcal{E} :: \Gamma\,;\Delta'\,;\Omega\cdot p\cdot\Omega' \Longrightarrow \gamma$.

$\Gamma\,;\Delta,\Delta'\,;\Omega \Longrightarrow \gamma$ cut on $\mathcal{D}'$ and $\mathcal{E}$

Subcase. $\mathcal{E} :: \Gamma\,;\Delta',p\,;\Omega \Longrightarrow \gamma$.

$\Gamma\,;\Delta,\Delta'\,;\Omega \Longrightarrow \gamma$ cut on $\mathcal{D}'$ and $\mathcal{E}$

Case. $\mathcal{D} = \dfrac{\mathcal{D}' :: \Gamma\,;\cdot\,;\cdot \Longrightarrow \cdot\,;p \quad p \text{ right-biased}}{\Gamma\,;\cdot \gg p}$ and $\mathcal{E} :: \Gamma,p\,;\Delta\,;\Omega \Longrightarrow \gamma$.

$\Gamma\,;\Delta\,;\Omega \Longrightarrow \gamma$ cut on $\mathcal{D}'$ and $\mathcal{E}$

Case. $\mathcal{D} :: \Gamma\,;\Delta\,;\Omega \Longrightarrow p\,;\cdot$ and $\mathcal{E} = \dfrac{\mathcal{E}' :: \Gamma\,;\Delta',p\,;\cdot \Longrightarrow \cdot\,;Q \quad p \text{ left-biased}}{\Gamma\,;\Delta'\,;p \ll Q}$

$\Gamma\,;\Delta,\Delta'\,;\Omega \Longrightarrow \cdot\,;Q$ cut on $\mathcal{D}$ and $\mathcal{E}'$

For commuting cuts, we commute into the available active derivation. There is no need to consider commuting a cut across a focus rule.

Left-commutative cuts. Where the cut formula is a side-formula on the left.

Case. The cut-formula A in the active zone. For instance,

$$\mathcal{E} = \frac{\mathcal{E}_1 :: \Gamma\,;\Delta\,;\Omega\cdot A \Longrightarrow B\,;\cdot \quad \mathcal{E}_2 :: \Gamma\,;\Delta\,;\Omega\cdot A \Longrightarrow C\,;\cdot}{\Gamma\,;\Delta\,;\Omega\cdot A \Longrightarrow B \,\&\, C\,;\cdot}$$

Subcase. $\mathcal{D} :: \Gamma\,;\Delta' \gg A$.

$\Gamma\,;\Delta,\Delta'\,;\Omega \Longrightarrow B\,;\cdot$ cut on $\mathcal{D}$ and $\mathcal{E}_1$

$\Gamma\,;\Delta,\Delta'\,;\Omega \Longrightarrow C\,;\cdot$ cut on $\mathcal{D}$ and $\mathcal{E}_2$

$\Gamma\,;\Delta,\Delta'\,;\Omega \Longrightarrow B \,\&\, C\,;\cdot$ $\&R$

Subcase. $\mathcal{D} :: \Gamma\,;\Delta'\,;\Omega' \Longrightarrow A\,;\cdot$ or $\mathcal{D} :: \Gamma\,;\Delta'\,;\Omega' \Longrightarrow \cdot\,;A$.

$\Gamma\,;\Delta,\Delta'\,;\Omega'\cdot\Omega \Longrightarrow B\,;\cdot$ cut on $\mathcal{D}$ and $\mathcal{E}_1$

$\Gamma\,;\Delta,\Delta'\,;\Omega'\cdot\Omega \Longrightarrow C\,;\cdot$ cut on $\mathcal{D}$ and $\mathcal{E}_2$

$\Gamma\,;\Delta,\Delta'\,;\Omega'\cdot\Omega \Longrightarrow B \,\&\, C\,;\cdot$ $\&R$

Case. The cut-formula is left-synchronous, and in the linear zone. For instance:

$$\mathcal{E} = \frac{\mathcal{E}_1 :: \Gamma\,;\Delta,A\,;\Omega \Longrightarrow B \quad \mathcal{E}_2 :: \Gamma\,;\Delta,A\,;\Omega \Longrightarrow C\,;\cdot}{\Gamma\,;\Delta,A\,;\Omega \Longrightarrow B \,\&\, C\,;\cdot}$$

Subcase. $\mathcal{D} :: \Gamma ; \Delta' ; \Omega' \Longrightarrow A ; \cdot$ or $\mathcal{D} :: \Gamma ; \Delta' ; \Omega' \Longrightarrow \cdot ; A$.

$\Gamma ; \Delta, \Delta' ; \Omega' \cdot \Omega \Longrightarrow B$	cut on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma ; \Delta, \Delta' ; \Omega' \cdot \Omega \Longrightarrow C$	cut on $\mathcal{D}$ and $\mathcal{E}_2$
$\Gamma ; \Delta, \Delta' ; \Omega' \cdot \Omega \Longrightarrow B \mathbin{\&} C$	$\&R$

Subcase. $\mathcal{E} :: \Gamma ; \Delta' \gg A$. As A is left-synchronous, it is either an atom or right-asynchronous.

Subcase $A = p(\vec{t})$ and left-biased. In this case $\mathcal{E} = \dfrac{}{\Gamma ; p(\vec{t}) \gg p(\vec{t})}$ rinit. Use rfoc to get $\Gamma ; p(\vec{t}) ; \cdot \Longrightarrow \cdot ; p(\vec{t})$, and we're in the previous case.

Subcase $A = p(\vec{t})$ and right-biased. In this case, the only rule that can have concluded $\mathcal{E}$ is rb*, so we have $\Gamma ; p(\vec{t}) ; \cdot \Longrightarrow p(\vec{t}) ; \cdot$. Once again, this is addressed by the previous case.

Subcase A is right-right-asynchronous. Like the previous case, the last rule in $\mathcal{E}$ must have been rb, so we have $\Gamma ; \Delta' ; \cdot \Longrightarrow A ; \cdot$, which is a previously addressed case.

Case. The cut formula A is in the unrestricted context; characteristic examples:

(a) $\mathcal{E}$ ends with a left-active rule, say:

$$\mathcal{D} :: \Gamma ; \cdot ; \cdot \Longrightarrow A ; \cdot \qquad \mathcal{E} = \frac{\mathcal{E}' :: \Gamma, A ; \Delta ; \Omega \cdot B \cdot C \cdot \Omega' \Longrightarrow Q}{\Gamma, A ; \Delta ; \Omega \cdot B \otimes C \cdot \Omega' \Longrightarrow Q}$$

$\Gamma ; \Delta ; \Omega \cdot B \cdot C \cdot \Omega' \Longrightarrow Q$	cut on $\mathcal{D}'$ and $\mathcal{E}$
$\Gamma ; \Delta ; \Omega \cdot B \otimes C \cdot \Omega' \Longrightarrow Q$	$\otimes R$

(b) $\mathcal{E}$ ends with a right-active rule, say:

$$\mathcal{D} :: \Gamma ; \cdot ; \cdot \Longrightarrow A ; \cdot \qquad \mathcal{E} = \frac{\mathcal{E}_1 :: \Gamma, A ; \Delta ; \Omega \Longrightarrow B \quad \mathcal{E}_2 :: \Gamma, A ; \Delta ; \Omega \Longrightarrow C}{\Gamma, A ; \Delta ; \Omega \Longrightarrow B \mathbin{\&} C}$$

$\Gamma ; \Delta ; \Omega \Longrightarrow B$	cut on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma ; \Delta ; \Omega \Longrightarrow C$	cut on $\mathcal{D}$ and $\mathcal{E}_2$
$\Gamma ; \Delta ; \Omega \Longrightarrow B \mathbin{\&} C$	$\&R$

(c) $\mathcal{D}$ ends in a right-focal rule, say:

$$\mathcal{D} :: \Gamma ; \cdot ; \cdot \Longrightarrow A ; \cdot \qquad \mathcal{E} = \frac{\mathcal{E}_1 :: \Gamma, A ; \Delta_1 \gg B \quad \mathcal{E}_2 :: \Gamma, A ; \Delta_2 \gg B}{\Gamma, A ; \Delta_1, \Delta_2 \gg B \otimes C}$$

$\Gamma ; \Delta_1 \gg B$	cut on $\mathcal{D}$ and $\mathcal{E}_1$
$\Gamma ; \Delta_2 \gg C$	cut on $\mathcal{D}$ and $\mathcal{E}_2$
$\Gamma ; \Delta_1, \Delta_2 \gg B \otimes C$	$\otimes R$

Right-commutative cuts. Where the cut formula is a side-formula on the right.

Case. $\mathcal{D}$ ends in a left-active rule, say:

$$\mathcal{D} = \frac{\mathcal{D}' :: \Gamma ; \Delta ; \Omega \cdot B \cdot C \cdot \Omega' \Longrightarrow A ; \cdot}{\Gamma ; \Delta ; \Omega \cdot B \otimes C \cdot \Omega' \Longrightarrow A ; \cdot}$$

Subcase. $\mathcal{E} :: \Gamma ; \Delta' ; \Omega'_A \cdot A \cdot \Omega'_A \Longrightarrow \gamma$.

$\Gamma ; \Delta, \Delta' ; \Omega_A \cdot \Omega \cdot B \cdot C \cdot \Omega' \cdot \Omega'_A \Longrightarrow \gamma$ — cut on $\mathcal{D}'$ and $\mathcal{E}$

$\Gamma ; \Delta, \Delta' ; \Omega_A \cdot \Omega \cdot B \otimes C \cdot \Omega' \cdot \Omega'_A \Longrightarrow \gamma$ — $\otimes L$.

Subcase. $\mathcal{E} :: \Gamma ; \Delta', A ; \Omega_A \Longrightarrow \gamma$.

$\Gamma ; \Delta, \Delta' ; \Omega_A \cdot \Omega \cdot B \cdot C \cdot \Omega' \Longrightarrow D$ — cut on $\mathcal{D}'$ and $\mathcal{E}$

$\Gamma ; \Delta, \Delta' ; \Omega_A \cdot \Omega \cdot B \otimes C \cdot \Omega' \Longrightarrow D$ — $\otimes L$

Subcase. Any case where A is in the unrestricted zone in the conclusion of $\mathcal{E}$ is impossible as there are some linear resources in the conclusion of $\mathcal{D}$.

Cases where the right hand side of the conclusion of $\mathcal{D}$ is of the form $\cdot ; A$ are similar.

Case. $\mathcal{D} :: \Gamma ; \Delta' \gg A$ is not a right-commutative case as A is not a side-formula in this derivation.

Case. The only remaining case is where the conclusion of $\mathcal{D}$ is a neutral sequent, i.e., $\mathcal{D} :: \Gamma ; \Delta ; \cdot \Longrightarrow \cdot ; A$ and A is right-synchronous. By the structure of sequents, it follows that A is right-synchronous. There are only two cases to consider.

Subcase A is a left-biased atom and the last rule in $\mathcal{D}$ is rfoc. In this case A is not a side-formula, so this is not a right-commutative case.

Subcase The last rule in $\mathcal{D}$ is lfoc:

$$\mathcal{D} = \frac{\mathcal{D}' :: \Gamma ; \Delta ; P \ll A}{\Gamma ; \Delta, P ; \cdot \Longrightarrow \cdot ; A}$$

In this case, the strategy is to permute the cut upwards in $\mathcal{E}$.

Subcase $\mathcal{E} :: \Gamma ; \Delta' ; \Omega \cdot A \cdot \Omega' \Longrightarrow \gamma$ and A is not the principal formula in the last rule in $\mathcal{E}$. For example:

$$\mathcal{E}' = \frac{\mathcal{E}'' :: \Gamma ; \Delta' ; \Omega \cdot D \otimes E \cdot \Omega' \cdot A \cdot \Omega'' \Longrightarrow \gamma}{\Gamma ; \Delta' ; \Omega \cdot D \otimes E \cdot \Omega' \cdot A \cdot \Omega'' \Longrightarrow \gamma}$$

$\Gamma ; \Delta, P, \Delta' ; \Omega \cdot D \cdot E \cdot \Omega' \cdot A \cdot \Omega'' \Longrightarrow \gamma$ — cut on $\mathcal{D}$ and $\mathcal{E}''$

$\Gamma ; \Delta, P, \Delta' ; \Omega \cdot D \otimes E \cdot \Omega' \cdot A \cdot \Omega'' \Longrightarrow \gamma$ — $\otimes L$

Subcase $\mathcal{E} :: \Gamma ; \Delta' ; \Omega \cdot A \cdot \Omega' \Longrightarrow \gamma$, A is the principal formula. If Ω and Ω' are not empty, or γ is of the form $C ; \cdot$, then there is a similar derivation where A is not the principal formula and we are back in the previous case. Thus, we only need to consider the case where $\mathcal{E} :: \Gamma ; \Delta' ; A \Longrightarrow \cdot ; Q$.

$\Gamma ; \Delta, \Delta' ; B \ll Q$ cut on $\mathcal{D}'$ and $\mathcal{E}$ (type 5)
$\Gamma ; \Delta, \Delta', B ; \cdot \Longrightarrow \cdot ; Q$ lfoc

Subcase $\mathcal{E} :: \Gamma ; \Delta', A ; \Omega \Longrightarrow \gamma$. If Ω is non-empty or γ is of the form $C ; \cdot$, then we are in a similar situation as the first subcase. Thus, the only interesting case is where $\mathcal{E} :: \Gamma' ; \Delta', A ; \cdot \Longrightarrow \cdot ; Q$.

$\Gamma ; \Delta, \Delta' ; B \ll A$ cut on $\mathcal{D}'$ and $\mathcal{E}$ (type 5)
$\Gamma ; \Delta, \Delta', B ; \cdot \Longrightarrow \cdot ; Q$ lfoc

Type 5 cuts. We have now completed the inventory of all cuts except those of type 5. For these cuts, we recurse into the first derivation $\mathcal{D} :: \Gamma ; \Delta ; B \ll A$. The following are the key cases.

Case $\mathcal{D}$ ends with lb or lb*. For example, for lb,

$$\mathcal{D} = \frac{\mathcal{D}' :: \Gamma ; \Delta ; L \Longrightarrow \cdot ; A}{\Gamma ; \Delta ; L \ll A} \qquad \mathcal{E} :: \Gamma ; \Delta' ; A \Longrightarrow \cdot ; Q$$

$\Gamma ; \Delta, \Delta' ; L \Longrightarrow \cdot ; Q$ cut on $\mathcal{D}'$ and $\mathcal{E}$ ($\mathcal{D}'$ smaller)
$\Gamma ; \Delta, \Delta' ; L \ll Q$ lb

The case for $\mathcal{E} :: \Gamma ; \Delta', A ; \cdot \Longrightarrow \cdot ; Q$ is similar.

Case $\mathcal{D}$ ends with a left-focal rule. For example:

$$\mathcal{D} = \frac{\mathcal{D}' :: \Gamma ; \Delta ; B \ll A}{\Gamma ; \Delta ; B \,\&\, C \ll A} \,\&L_1 \qquad \mathcal{E} :: \Gamma ; \Delta' ; A \Longrightarrow \cdot ; Q$$

$\Gamma ; \Delta, \Delta' ; B \ll Q$ cut on $\mathcal{D}'$ and $\mathcal{E}$ (type 5, with $\mathcal{D}'$ smaller)
$\Gamma ; \Delta, \Delta' ; B \,\&\, C \ll Q$ $\&L_1$

Once again, the case for $\mathcal{E} :: \Gamma ; \Delta', A ; \cdot \Longrightarrow \cdot ; Q$ is similar. □

We will use the cut theorem to show that all rules of the non-focusing calculus are admissible in the focusing calculus by interpreting the non-focusing sequents as active sequents. To achieve this, we first need the equivalent of the identity principle for the focusing calculus: $\Gamma ; \cdot ; A \Longrightarrow A ; \cdot$. In the focusing calculus this is not a straightforward induction because of the occurrence restrictions on focal sequents. To illustrate, Δ in $\Gamma ; \Delta \gg A$ cannot contain any left-asynchronous propositions, so the proof of $\Gamma ; \cdot ; A \otimes B \Longrightarrow A \otimes B ; \cdot$ is not simply a proof of $\Gamma ; A \otimes B \gg A \otimes B$. We generalise the induction by furnishing a proof in terms of an *expansion* of these asynchronous propositions.

Definition 6.8 (Expansion)**.**

1. *The* left-expansion *of a proposition A, written* $\mathrm{lexp}(A)$*, is a set of two-zoned contexts defined inductively by the following equations.*

$$\begin{aligned}
\mathrm{lexp}(P) &= \{(\cdot ; P)\} \\
\mathrm{lexp}(A \otimes B) &= \{(\Gamma_A \cup \Gamma_B ; \Delta_A, \Delta_B) \ :\ (\Gamma_A ; \Delta_A) \in \mathrm{lexp}(A) \textit{ and } (\Gamma_B ; \Delta_B) \in \mathrm{lexp}(B)\} \\
\mathrm{lexp}(\mathbf{1}) &= \{(\cdot ; \cdot)\} \\
\mathrm{lexp}(A \oplus B) &= \mathrm{lexp}(A) \cup \mathrm{lexp}(B) \\
\mathrm{lexp}(\mathbf{0}) &= \emptyset \\
\mathrm{lexp}(!\,A) &= \{A ; \cdot\} \\
\mathrm{lexp}(\exists x.A) &= \mathrm{lexp}([a/x]A)
\end{aligned}$$

2. *The* right-expansion *of a proposition A, written* $\mathrm{rexp}(A)$*, is a set of elements of the form $\Gamma ; \Delta \Longrightarrow Q$ defined inductively by the following equations.*

$$\begin{aligned}
\mathrm{rexp}(Q) &= \{(\cdot ; \cdot \Longrightarrow Q)\} \\
\mathrm{rexp}(A \mathbin{\&} B) &= \mathrm{rexp}(A) \cup \mathrm{rexp}(B) \\
\mathrm{rexp}(\top) &= \emptyset \\
\mathrm{rexp}(A \multimap B) &= \left\{ (\Gamma_A \cup \Gamma_B ; \Delta_A, \Delta_B \Longrightarrow Q) \ :\ \begin{array}{l} (\Gamma_A ; \Delta_A) \in \mathrm{lexp}(A) \textit{ and} \\ (\Gamma_B ; \Delta_B \Longrightarrow Q) \in \mathrm{rexp}(B) \end{array} \right\} \\
\mathrm{rexp}(\forall x.A) &= \mathrm{rexp}([a/x]A)
\end{aligned}$$

This definition is associated with a key *expansion lemma.*

Lemma 6.9 (Expansion lemma)**.** *For any proposition A:*

1. *For any Γ, Δ, Ω and γ,*
   if for every $(\Gamma' ; \Delta') \in \mathrm{lexp}(A)$, the sequent $\Gamma, \Gamma' ; \Delta, \Delta' ; \Omega \Longrightarrow \gamma$ is derivable,
   then $\Gamma ; \Delta ; \Omega \cdot A \Longrightarrow \gamma$.

2. *For any* Γ, Δ *and* Ω,
   if for every $(\Gamma' ; \Delta' \Longrightarrow Q') \in \mathrm{rexp}(A)$*, the sequent* $\Gamma, \Gamma' ; \Delta, \Delta' ; \Omega \Longrightarrow \cdot ; Q'$ *is derivable, then* $\Gamma ; \Delta ; \Omega \Longrightarrow A ; \cdot$.

Proof. By induction on the structure of A. We present here some of the key cases.

Case of A is left-asynchronous, say $B \otimes C$, and arguing for $\mathrm{lexp}(A)$. Let Γ, Δ, Ω and γ be given and assume that for every $(\Gamma' ; \Delta') \in \mathrm{lexp}(B \otimes C)$, $\Gamma, \Gamma' ; \Delta, \Delta' ; \Omega \Longrightarrow \gamma$. Choose such a $(\Gamma' ; \Delta') \in \mathrm{lexp}(A \otimes B)$. By definition 6.8, $(\Gamma' ; \Delta')$ has the form $(\Gamma'_B \cup \Gamma'_C ; \Delta'_B, \Delta'_C)$ such that $(\Gamma'_B ; \Delta'_B) \in \mathrm{lexp}(B)$ and $(\Gamma'_C ; \Delta'_C) \in \mathrm{lexp}(C)$.

$\Gamma, \Gamma' \setminus \Gamma'_B ; \Delta ; \Omega \cdot B \Longrightarrow \gamma$	i.h. for B, $(\Gamma, \Gamma' \setminus \Gamma'_B)$, Δ, and Ω
$\Gamma ; \Delta ; \Omega \cdot B \cdot C \Longrightarrow \gamma$	i.h. for C, Γ, Δ and $(\Omega \cdot B)$
$\Gamma ; \Delta ; \Omega \cdot B \otimes C \Longrightarrow \gamma$	$\otimes L$

Then we note that this conclusion is independent of the choice of $(\Gamma' ; \Delta')$. Other cases of $\mathrm{lexp}(A)$ with A being left-asynchronous have similar arguments.

Case of $A = P$ and arguing for $\mathrm{lexp}(A)$. In this case, any $(\Gamma' ; \Delta') \in \mathrm{lexp}(A)$ has the form $(\cdot ; P)$.

$\Gamma ; \Delta, P ; \Omega \Longrightarrow \gamma$	assumption
$\Gamma ; \Delta ; \Omega \cdot P \Longrightarrow \gamma$	lact

This completes the inventory of cases for lexp.

Case of $A = B \,\&\, C$ and arguing for $\mathrm{rexp}(A)$. Let Γ, Δ and Ω be given and assume that for every $(\Gamma' ; \Delta' \Longrightarrow Q') \in \mathrm{lexp}(B \,\&\, C)$, $\Gamma, \Gamma' ; \Delta, \Delta' ; \Omega \Longrightarrow \cdot ; Q'$. By definition 6.8, $\mathrm{lexp}(B \otimes C) = \mathrm{lexp}(B) \cup \mathrm{lexp}(C)$ the outer quantification also holds for each component of the union; *i.e.*, for every $(\Gamma' ; \Delta' \Longrightarrow Q') \in \mathrm{lexp}(B)$, $\Gamma, \Gamma' ; \Delta, \Delta' ; \Omega \Longrightarrow \cdot ; Q'$, and similarly for $\mathrm{lexp}(C)$.

$\Gamma ; \Delta ; \Omega \Longrightarrow B ; \cdot$	i.h. on B, Γ, Δ and Ω
$\Gamma ; \Delta ; \Omega \Longrightarrow C ; \cdot$	i.h. on C, Γ, Δ and Ω
$\Gamma ; \Delta ; \Omega \Longrightarrow B \,\&\, C ; \cdot$	$\& R$

Other cases for $\mathrm{rexp}(A)$ with A being right-asynchronous have similar arguments.

Case of $A = Q$ and arguing for $\mathrm{rexp}(A)$. In this case, all $(\Gamma' ; \Delta' \Longrightarrow Q') \in \mathrm{rexp}(A)$ have the form $(\cdot ; \cdot \Longrightarrow Q)$.

$\Gamma\,;\Delta\,;\Omega \Longrightarrow \cdot\,; Q$ assumption

$\Gamma\,;\Delta\,;\Omega \Longrightarrow Q\,;\cdot$ ract

This completes the inventory of all cases for rexp(A). □

We use the expansion lemma to establish the key theorem that will give us the identity principle as a corollary.

Theorem 6.10.

1. *For any proposition A, for every* $(\Gamma\,;\Delta) \in \text{lexp}(A)$*, we can show* $\Gamma\,;\Delta \gg A$.
2. *For any proposition A, for every* $(\Gamma\,;\Delta \Longrightarrow Q) \in \text{rexp}(A)$*, we can show* $\Gamma\,;\Delta\,;A \ll Q$.

Proof. By structural induction on A and the definition of lexp and rexp (defn. 6.8). In the inductive argument, the case for rexp(Q) where Q is non-atomic can be used in the argument for lexp(A) (and lexp(P) for rexp(A) similarly). This order is well-founded because there are only finitely many phase changes between synchronous and asynchronous subformulas in a given proposition. We show below some of the key cases of the induction.

Case of lexp($A \otimes B$): every $(\Gamma\,;\Delta) \in \text{lexp}(A \otimes B)$ is of the form $(\Gamma_A \cup \Gamma_B\,;\Delta_A,\Delta_B)$ where $(\Gamma_A\,;\Delta_A) \in \text{lexp}(A)$ and $(\Gamma_B\,;\Delta_B) \in \text{lexp}(B)$.

$\Gamma_A\,;\Delta_A \gg A$ i.h.

$\Gamma_A \cup \Gamma_B\,;\Delta_A \gg A$ weakening (thm. 6.3)

$\Gamma_A \cup \Gamma_B\,;\Delta_B \gg B$ similarly

$\Gamma_A \cup \Gamma_B\,;\Delta_A,\Delta_B \gg A \otimes B$ $\otimes R$.

All inductive cases of lexp are similar.

Case of rexp($A \,\&\, B$): let $(\Gamma\,;\Delta \Longrightarrow Q) \in \text{rexp}(A \,\&\, B)$ be given. By defn 6.8, we have (without loss of generality), $(\Gamma\,;\Delta \Longrightarrow Q) \in \text{rexp}(A)$.

$\Gamma\,;\Delta\,;A \ll Q$ i.h.

$\Gamma\,;\Delta\,;A \,\&\, B \ll Q$ $\& L_1$.

The other inductive cases of rexp are similar.

Case of lexp(Q): There are three sub-cases here.

Subcase Q is a left-biased atomic proposition $p(\vec{t})$. By rinit, $\cdot\,; p(\vec{t}) \gg p(\vec{t})$.

Subcase Q is a right-biased atomic proposition $p(\vec{t})$.

$\cdot\,;\cdot\,;p(\vec{t}) \ll p(\vec{t})$	linit
$\cdot\,;p(\vec{t})\,;\cdot \Longrightarrow \cdot\,;p(\vec{t})$	lfoc
$\cdot\,;p(\vec{t}) \gg p(\vec{t})$	rb*

Subcase Q is a non-atomic proposition. Because it is left-synchronous, it is right-asynchronous.

For every $(\Gamma\,;\Delta \Longrightarrow Q') \in \mathrm{rexp}(Q), \Gamma\,;\Delta\,;Q \ll Q'$	i.h. (type 2)
For every $(\Gamma\,;\Delta \Longrightarrow Q') \in \mathrm{rexp}(Q), \Gamma\,;\Delta, Q\,;\cdot \Longrightarrow \cdot\,;Q'$	lfoc
$\cdot\,;Q\,;\cdot \Longrightarrow \cdot\,;Q$	expansion lemma (lem. 6.9)
$\cdot\,;Q\,;\cdot \Longrightarrow Q\,;\cdot$	ract
$\cdot\,;Q \gg Q$	rb

The case of rexp(P) is similar. □

Corollary 6.11 (Identity principle)**.**
For any proposition A, the sequent $\cdot\,;\cdot\,;A \Longrightarrow A\,;\cdot$ is derivable.

Proof. Suppose A is right-synchronous, i.e., of the form Q. There are three cases here.

Case A is a right-biased atomic proposition $p(\vec{t})$.

$\cdot\,;\cdot\,;p(\vec{t}) \ll p(\vec{t})$	linit
$\cdot\,;p(\vec{t})\,;\cdot \Longrightarrow \cdot\,;p(\vec{t})$	lfoc
$\cdot\,;\cdot\,;p(\vec{t}) \Longrightarrow p(\vec{t})\,;\cdot$	lact and ract

Case A is a left-biased atomic proposition $p(\vec{t})$.

$\cdot\,;p(\vec{t}) \gg p(\vec{t})$	rinit
$\cdot\,;p(\vec{t})\,;\cdot \Longrightarrow \cdot\,;p(\vec{t})$	rfoc
$\cdot\,;\cdot\,;p(\vec{t}) \Longrightarrow p(\vec{t})\,;\cdot$	lact and ract

Case A is a non-atomic.

For every $(\Gamma\,;\Delta) \in \mathrm{lexp}(A), \Gamma\,;\Delta \gg A$	theorem 6.10
For every $(\Gamma\,;\Delta) \in \mathrm{lexp}(A), \Gamma\,;\Delta\,;\cdot \Longrightarrow \cdot\,;A$	rfoc

Note that A is non-atomic and right-synchronous, hence focile.

$\cdot\,;\cdot\,;A \Longrightarrow \cdot\,;A$	the expansion lemma (lem. 6.9)
$\cdot\,;\cdot\,;A \Longrightarrow A\,;\cdot$	ract

The case of A being left-synchronous has a similar argument. □

This specific statement of the identity principle will not be used in the completeness proof below; instead, we will use a slightly variant formulation.

Lemma 6.12. *The following are derivable (for arbitrary* Γ*,* A *and* B*):*

1. $\cdot\,;\cdot\,;A \cdot B \Longrightarrow A \otimes B\,;\cdot$
2. $\cdot\,;\cdot\,;\cdot \Longrightarrow \mathbf{1}\,;\cdot$
3. $\cdot\,;\cdot\,;A \Longrightarrow A \oplus B\,;\cdot$ *and* $\Gamma\,;\cdot\,;B \Longrightarrow A \oplus B\,;\cdot$
4. $A\,;\cdot\,;\cdot \Longrightarrow\,!A\,;\cdot$
5. $\cdot\,;\cdot\,;[a/x]A \Longrightarrow \exists x.A\,;\cdot$
6. $\cdot\,;\cdot\,;A \,\&\, B \Longrightarrow A\,;\cdot$ *and* $\Gamma\,;\cdot\,;A \,\&\, B \Longrightarrow B\,;\cdot$
7. $\cdot\,;\cdot\,;A \cdot A \multimap B \Longrightarrow B\,;\cdot$
8. $\cdot\,;\cdot\,;\forall x.A \Longrightarrow [a/x]A\,;\cdot$

Proof. Each case is a simple consequence of the identity principle (cor. 6.11). The following is a representative case for $A \otimes B$.

$\cdot\,;\cdot\,;A \otimes B \Longrightarrow A \otimes B\,;\cdot$ cor. 6.11.

There are two rules that can conclude this sequent: ract or $\otimes L$. In the former case

$\cdot\,;\cdot\,;A \otimes B \Longrightarrow \cdot\,;A \otimes B$ assumption
$\cdot\,;\cdot\,;A \cdot B \Longrightarrow \cdot\,;A \otimes B$ premiss of $\otimes L$ (only possible rule)
$\cdot\,;\cdot\,;A \cdot B \Longrightarrow A \otimes B\,;\cdot$ ract

In the latter case, the premiss is already of the required form $\cdot\,;\cdot\,;A \cdot B \Longrightarrow A \otimes B\,;\cdot$ The remaining cases use similar arguments. □

Theorem 6.13 (Completeness).
If $\Gamma\,;\Delta \Longrightarrow C$ *and* Ω *is any serialisation of* Δ*, then* $\Gamma\,;\cdot\,;\Omega \Longrightarrow C\,;\cdot$*.*

Proof. First we show that all ordinary rules are admissible in the focusing system using cut. We then proceed by induction on derivation $\mathcal{D} :: \Gamma\,;\Delta \Longrightarrow C$, splitting cases on the last applied rule, using cut and lemmas 6.4 and 6.12 as required. The following is a representative case for $\otimes R$:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma\,;\Delta \Longrightarrow A \quad \mathcal{D}_2 :: \Gamma\,;\Delta_2 \Longrightarrow B}{\Gamma\,;\Delta,\Delta' \Longrightarrow A \otimes B}\,\otimes R$$

Let Ω and Ω' be serialisations of Δ and Δ' respectively.

$$
\begin{array}{lr}
\Gamma\,;\cdot\,;\Omega \Longrightarrow A\,;\cdot & \text{i.h. on } \mathcal{D}_1 \\
\Gamma\,;\cdot\,;\Omega' \Longrightarrow B\,;\cdot & \text{i.h. on } \mathcal{D}_2 \\
\Gamma\,;\cdot\,;A\cdot B \Longrightarrow A\otimes B\,;\cdot & \text{lem. 6.12 and weakening (thm. 6.3)} \\
\Gamma\,;\cdot\,;\Omega\cdot\Omega' \Longrightarrow A\otimes B\,;\cdot & \text{cut twice}
\end{array}
$$

Any serialisation of Δ, Δ' is a permutation of $\Omega \cdot \Omega'$. □

As a remark, once we have the cut and the identity principle, the proof of completeness is extremely straightforward. There are other proofs of completeness of focusing calculi in the literature that do not use cut-elimination as a basis. Andreoli's original proof of completeness for a classical focusing calculus in [7] used a number of permutation arguments for rules. Howe's extension of focusing to intuitionistic and linear logics divided each case of Andreoli's permutation argument into a number of lemmas [57]. Each of Howe's lemma actually bears a strong resemblance to one of the commutative cases of cut, though a precise correspondence is hard to state given the dissimilarities of the two calculi. We believe that cut and identity—independent of their use in proving completeness—are sufficiently interesting in and of themselves as they substantiate the logical basis of focusing. Similar notions of cut and cut-admissibility also exist in Ludics [44], though our calculus and Ludics are philosophically dissimilar enough that we cannot simply import the cut-admissibility argument from Ludics. Rather, we choose to view our proof of cut-admissibility as belonging to a different tradition which sometimes goes by the keyphrase "structural cut-eliminiation" [92].

6.2 Forward focusing

As mentioned earlier, the primary benefit of focusing is the ability to generate derived "big step" inference rules: the intermediate results of a focusing or active phase are not important. Andreoli called these rules "bipoles" because they combine two phases with principal formulas of opposite polarities. Each derived rule starts (at the bottom) with a neutral sequent from which a synchronous proposition is selected for focus. This is followed by a sequence of focusing steps until the proposition under focus becomes asynchronous. Then, the active rules are applied, and we eventually obtain a collection of neutral sequents as the leaves of this fragment of the focused derivation. These neutral sequents are then treated as the premisses of the derived rule that produces the neutral sequent with which we started.

The derived rule calculus will be formally presented in sec. 6.2.2; here, we will motivate it with an example. Consider the negative proposition $q \otimes n \multimap d \otimes d \otimes d$[3] in the unrestricted context Γ. We start with focus on this proposition, and construct the following derivation tree.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma ; \Delta_1 \Longrightarrow q}{\Gamma ; \Delta_1 ; \cdot \Longrightarrow q ; \cdot}}{\Gamma ; \Delta_1 \gg q}\,\text{rb} \qquad \dfrac{\dfrac{\Gamma ; \Delta_2 \Longrightarrow n}{\Gamma ; \Delta_2 ; \cdot \Longrightarrow n ; \cdot}}{\Gamma ; \Delta_2 \gg n}\,\text{rb}}{\Gamma ; \Delta_1, \Delta_2 \gg q \otimes n}\,{\otimes}R \qquad \dfrac{\dfrac{\Gamma ; \Delta_3, d, d, d \Longrightarrow \cdot ; Q}{\Gamma ; \Delta_3 ; d \otimes d \otimes d \Longrightarrow \cdot ; Q}\,{\otimes}L;{\otimes}L;\text{lact} \times 3}{\Gamma ; \Delta_3 ; d \otimes d \otimes d \ll Q}\,\text{lb}}{\Gamma ; \Delta_1, \Delta_2, \Delta_3 ; q \otimes n \multimap d \otimes d \otimes d \ll Q}\,{\multimap}L}{\Gamma ; \Delta_1, \Delta_2, \Delta_3 \Longrightarrow Q}\,\text{copy}$$

Here we assume that all atoms are right-biased, so none of the branches of the derivation can be closed off with an “init” rule. Thus, we obtain the derived rule:

$$\dfrac{\Gamma ; \Delta_1 \Longrightarrow q \quad \Gamma ; \Delta_2 \Longrightarrow n \quad \Gamma ; \Delta_3, d, d, d \Longrightarrow Q}{\Gamma ; \Delta_1, \Delta_2, \Delta_3 \Longrightarrow Q} \tag{D_1}$$

The situation is considerably different if we assume that all atoms are left-biased. In this case, we get the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma ; q \gg q}\,\text{linit} \qquad \dfrac{}{\Gamma ; n \gg n}\,\text{linit}}{\Gamma ; q, n \gg q \otimes n}\,{\otimes}R \qquad \dfrac{\dfrac{\Gamma ; \Delta, d, d, d \Longrightarrow \cdot ; Q}{\Gamma ; \Delta ; d \otimes d \otimes d \Longrightarrow \cdot ; Q}\,{\otimes}L;{\otimes}L;\text{lact} \times 3}{\Gamma ; \Delta ; d \otimes d \otimes d \ll Q}\,\text{lb}}{\Gamma ; q, n, \Delta ; q \otimes n \multimap d \otimes d \otimes d \ll Q}\,{\multimap}L}{\Gamma ; q, n, \Delta \Longrightarrow Q}\,\text{copy}$$

In this left-biased case, we can terminate the left branch of the derivation with a pair of “init” rules. This rule forces the linear context in this branch of the proof to contain just the atoms q and n. The derived rule we obtain is, therefore,

$$\dfrac{\Gamma ; \Delta, d, d, d \Longrightarrow Q}{\Gamma ; \Delta, q, n \Longrightarrow Q} \tag{D_2}$$

There are two key differences to observe between the derived rules (D_1) and (D_2). The first is that simply altering the bias of the atoms has a huge impact on the kinds of rules

[3]This is the same change-machine example from sec. 3.2.1

that are generated; even if we completely ignore the semantic aspect, the rule (D_2) is vastly preferable to (D_1) because it is much easier to use single premiss rules.

The second — and more important — observation is that the rule that was generated for the left-biased atoms has a stronger and more obvious similarity to the proposition $q \otimes n \multimap d \otimes d \otimes d$ that was under focus. If we view the linear zone as the "state" of a system, then the rule (D_2) corresponds to transforming a portion of the state by replacing q and n by three ds (reading the rule from bottom to top). If, as is common for linear logic, the unrestricted context Γ contains state transition rules for some encoding of a stateful system, then the derived rules generated by left-biasing allows us to directly observe the evolution of the state of the system by looking at the composition of the linear zone.

We now construct the forward version of the focusing calculus. The general design is that intermediate sequents in the eager active and focusing phases are not be stored in any sequent database; instead, all sequents constructed during search are neutral sequents at the phase boundaries. This is achieved by first precomputing the derived rules that correspond to the *frontier literals* (see defn. 6.24) of the goal sequent.

6.2.1 Backward derived rules

For any given proposition, we are interested in constructing a derived inference for the proposition corresponding to a single pair of focusing and inverse phases. There are, however, important differences between backward reasoning bipoles and their forward analogue. As shown in the proofs of completeness for forward calculi (for example, theorem 3.11), forward sequents generally have fewer components than backward sequents; as forward rules have tight matching criteria, a stronger sequent will often fail to match an inference rule. The intent of this section is to transfer the idea of bipoles to forward derived rules. The details, particularly the proof of completeness (thm. 6.23), turn out to be surprisingly subtle, so for presentation purposes we recall the backward construction of bipoles.

The essential idea is to interpret a proposition itself as the (derived) rules that it embodies. Every proposition is viewed as a relation between the conclusion of the rule and its premisses at the leaves of the bipole. Both the conclusion and the premisses of this bipole are neutral sequents, which we indicate by means of a double-headed sequent

right-focal

$$\frac{p \text{ left-biased}}{\mathsf{foc}^{+}_{\Uparrow}(p)[\Gamma\,;p \Longrightarrow\!\!\!\!\Rightarrow \cdot] \hookrightarrow \cdot}\ \text{rinit}$$

$$\frac{\mathsf{foc}^{+}_{\Uparrow}(A)[\Gamma\,;\Delta_1 \Longrightarrow\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Sigma_1 \qquad \mathsf{foc}^{+}_{\Uparrow}(A)[\Gamma\,;\Delta_2 \Longrightarrow\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Sigma_2}{\mathsf{foc}^{+}_{\Uparrow}(A \otimes B)[\Gamma\,;\Delta_1, \Delta_2 \Longrightarrow\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Sigma_1 \cdot \Sigma_2}\ \otimes F \qquad \frac{}{\mathsf{foc}^{+}_{\Uparrow}(\mathbf{1})[\Gamma\,;\cdot \Longrightarrow\!\!\!\!\Rightarrow \cdot] \hookrightarrow \cdot}\ \mathbf{1}F$$

$$\frac{\mathsf{foc}^{+}_{\Uparrow}(A_i)[s] \hookrightarrow \Sigma}{\mathsf{foc}^{+}_{\Uparrow}(A_1 \oplus A_2)[s] \hookrightarrow \Sigma}\ \oplus F_i \qquad \frac{\mathsf{act}_{\Uparrow}(\cdot\,;\cdot\,;\cdot \Longrightarrow A)[\Gamma\,;\cdot \Longrightarrow\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Sigma}{\mathsf{foc}^{+}_{\Uparrow}(!\,A)[\Gamma\,;\cdot \Longrightarrow\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Sigma}\ !\,F$$

$$\frac{\mathsf{foc}^{+}_{\Uparrow}([t/x]A)[\Gamma\,;\Delta \Longrightarrow\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Sigma}{\mathsf{foc}^{+}_{\Uparrow}(\exists x.A)[\Gamma\,;\Delta \Longrightarrow\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Sigma}\ \exists F \qquad \frac{\mathsf{act}_{\Uparrow}(\cdot\,;\cdot\,;\cdot \Longrightarrow R)[s] \hookrightarrow \Sigma}{\mathsf{foc}^{+}_{\Uparrow}(R)[s] \hookrightarrow \Sigma}\ FA^{+}$$

left-focal

$$\frac{p \text{ right-biased}}{\mathsf{foc}^{-}_{\Uparrow}(p)[\Gamma\,;\cdot \Longrightarrow\!\!\!\!\Rightarrow p] \hookrightarrow \cdot}\ \text{linit} \qquad \frac{\mathsf{foc}^{-}_{\Uparrow}(A_i)[s] \hookrightarrow \Sigma}{\mathsf{foc}^{-}_{\Uparrow}(A_1 \,\&\, A_2)[s] \hookrightarrow \Sigma}\ \& F_i$$

$$\frac{\mathsf{foc}^{-}_{\Uparrow}(B)[\Gamma\,;\Delta_1 \Longrightarrow\!\!\!\!\Rightarrow Q] \hookrightarrow \Sigma_1 \qquad \mathsf{foc}^{+}_{\Uparrow}(A)[\Gamma\,;\Delta_2 \Longrightarrow\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Sigma_2}{\mathsf{foc}^{-}_{\Uparrow}(A \multimap B)[\Gamma\,;\Delta_1, \Delta_2 \Longrightarrow\!\!\!\!\Rightarrow Q] \hookrightarrow \Sigma_1 \cdot \Sigma_2}\ \multimap F \qquad \frac{\mathsf{foc}^{-}_{\Uparrow}([t/x]A)[\Gamma\,;\Delta \Longrightarrow\!\!\!\!\Rightarrow Q] \hookrightarrow \Sigma}{\mathsf{foc}^{-}_{\Uparrow}(\forall x.A)[\Gamma\,;\Delta \Longrightarrow\!\!\!\!\Rightarrow Q] \hookrightarrow \Sigma}\ \forall F$$

$$\frac{\mathsf{act}_{\Uparrow}(\cdot\,;\cdot\,;L \Longrightarrow \cdot)[s] \hookrightarrow \Sigma}{\mathsf{foc}^{-}_{\Uparrow}(L)[s] \hookrightarrow \Sigma}\ FA^{-}$$

Figure 6.2: backward derived rules: focal phase

arrow ($\Longrightarrow\!\!\!\!\Rightarrow$). Given a neutral conclusion $\Gamma\,;\Delta \Longrightarrow\!\!\!\!\Rightarrow Q$, one proposition from Γ, Δ or Q is selected for focus, and the relational interpretation of the conclusion with respect to the selected proposition provides the new (neutral) premisses of the bipole.

There are three important classes of these relational interpretations:

1. Right focal relations for the focus formula A, written $\mathsf{foc}^{+}_{\Uparrow}(A)$.
2. Left focal relations for the focus formula A, written $\mathsf{foc}^{-}_{\Uparrow}(A)$.
3. Active relations, written $\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow \xi)$, where ξ is either $\cdot$ or a proposition C.

Each relation R takes as input the conclusion sequent s, and produces a sequence of premiss sequents $\Sigma = s_1 \cdot s_2 \cdots s_n$; we write this as $R[s] \hookrightarrow \Sigma$.

These relations are defined in fig. 6.2 and 6.3. The focal relations are understood as defining derived rules corresponding to a given proposition. If in a neutral sequent

active

$$\frac{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow A)[s] \hookrightarrow \Sigma_1 \quad \mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow B)[s] \hookrightarrow \Sigma_2}{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow A \mathbin{\&} B)[s] \hookrightarrow \Sigma_1 \cdot \Sigma_2}\;\&A$$

$$\frac{}{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow \top)[s] \hookrightarrow \cdot}\;\top A$$

$$\frac{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot A \cdot \Omega' \Longrightarrow \xi)[s] \hookrightarrow \Sigma_1 \quad \mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot B \cdot \Omega' \Longrightarrow \xi)[s] \hookrightarrow \Sigma_2}{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot A \oplus B \cdot \Omega' \Longrightarrow \xi)[s] \hookrightarrow \Sigma_1 \cdot \Sigma_2}\;\oplus A$$

$$\frac{}{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot \mathbf{0} \Longrightarrow \xi)[s] \hookrightarrow \cdot}\;\mathbf{0}A$$

$$\frac{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot A \cdot B \cdot \Omega' \Longrightarrow \xi)[s] \hookrightarrow \Sigma}{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot A \otimes B \cdot \Omega' \Longrightarrow \xi)[s] \hookrightarrow \Sigma}\;\otimes A \qquad \frac{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot \Omega' \Longrightarrow \xi)[s] \hookrightarrow \Sigma}{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot \mathbf{1} \cdot \Omega' \Longrightarrow \xi)[s] \hookrightarrow \Sigma}\;\mathbf{1}A$$

$$\frac{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot A \cdot \Omega' \Longrightarrow B)[s] \hookrightarrow \Sigma}{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot \Omega' \Longrightarrow A \multimap B)[s] \hookrightarrow \Sigma}\;\multimap A \qquad \frac{\mathsf{act}_{\Uparrow}(\Gamma, A\,;\Delta\,;\Omega \cdot \Omega' \Longrightarrow \xi)[s] \hookrightarrow \Sigma}{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot {!}A \cdot \Omega' \Longrightarrow \xi)[s] \hookrightarrow \Sigma}\;{!}A$$

$$\frac{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow [a/x]A)[s] \hookrightarrow \Sigma}{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow \forall x.A)[s] \hookrightarrow \Sigma}\;\forall R^a \qquad \frac{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot [a/x]A \cdot \Omega' \Longrightarrow \xi)[s] \hookrightarrow \Sigma}{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot \exists x.A \cdot \Omega' \Longrightarrow \xi)[s] \hookrightarrow \Sigma}\;\exists L^a$$

$$\frac{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta, P\,;\Omega \cdot \Omega' \Longrightarrow \xi)[s] \hookrightarrow \Sigma}{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\Omega \cdot P \cdot \Omega' \Longrightarrow \xi)[s] \hookrightarrow \Sigma}\;\text{bact}$$

$$\frac{}{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\cdot \Longrightarrow \cdot)[\Gamma'\,;\Delta' \Longrightarrow\!\!\!\!\!\Rightarrow Q] \hookrightarrow \Gamma, \Gamma'\,;\Delta, \Delta' \Longrightarrow\!\!\!\!\!\Rightarrow Q}\;\text{match}$$

$$\frac{}{\mathsf{act}_{\Uparrow}(\Gamma\,;\Delta\,;\cdot \Longrightarrow Q)[\Gamma'\,;\Delta' \Longrightarrow\!\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Gamma, \Gamma'\,;\Delta, \Delta' \Longrightarrow\!\!\!\!\!\Rightarrow Q}\;\text{match}'$$

Figure 6.3: backward derived rules: active phase

$\Gamma\,;\Delta \Longrightarrow\!\!\!\!\!\Rightarrow Q$ we focus on the right, then $\mathsf{foc}^{+}_{\Uparrow}(Q)$ would relate this sequent to the possible premisses in the entire bipole.

$$\frac{\left(\mathsf{foc}^{+}_{\Uparrow}(Q)[\Gamma\,;\Delta \Longrightarrow\!\!\!\!\!\Rightarrow \cdot] \hookrightarrow s_1 \cdot s_2 \cdots s_n\right) \quad s_1 \quad s_2 \quad \cdots \quad s_n}{\Gamma\,;\Delta \Longrightarrow\!\!\!\!\!\Rightarrow Q}\;\mathsf{foc}^{+}_{\Uparrow}$$

Similarly for $\mathsf{foc}^{-}_{\Uparrow}$ we have two rules:

$$\frac{\left(\mathsf{foc}^{-}_{\Uparrow}(P)[\Gamma\,;\Delta \Longrightarrow\!\!\!\!\!\Rightarrow Q] \hookrightarrow s_1 \cdot s_2 \cdots s_n\right) \quad s_1 \quad s_2 \quad \cdots \quad s_n}{\Gamma\,;\Delta, P \Longrightarrow\!\!\!\!\!\Rightarrow Q}\;\mathsf{foc}^{-}_{\Uparrow}$$

$$\frac{\left(\mathsf{foc}^{-}_{\Uparrow}(A)[\Gamma, A\,;\Delta \Longrightarrow\!\!\!\!\!\Rightarrow Q] \hookrightarrow s_1 \cdot s_2 \cdots s_n\right) \quad s_1 \quad s_2 \quad \cdots \quad s_n}{\Gamma, A\,;\Delta \Longrightarrow\!\!\!\!\!\Rightarrow Q}\;{!}\,\mathsf{foc}^{-}_{\Uparrow}$$

Theorem 6.14 (soundness). *Say that* $\Gamma\,;\Delta \Longrightarrow\!\!\!\!\!\Rightarrow Q$ *is* sound *if* $\Gamma\,;\Delta\,;\cdot \Longrightarrow \cdot\,;Q$ *is derivable.*

1. *If* $\mathsf{foc}^+_\Uparrow(A)[\Gamma\,;\Delta \Longrightarrow\!\!\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Sigma$ *and* Σ *are sound, then* $\Gamma\,;\Delta \gg A$.
2. *If* $\mathsf{foc}^-_\Uparrow(A)[\Gamma\,;\Delta \Longrightarrow\!\!\!\!\!\!\Rightarrow Q] \hookrightarrow \Sigma$ *and* Σ *are sound, then* $\Gamma\,;\Delta\,;A \ll Q$.
3. *If* $\mathsf{act}_\Uparrow(\Gamma\,;\Delta\,;\Omega \Longrightarrow \cdot)[\Gamma'\,;\Delta' \Longrightarrow\!\!\!\!\!\!\Rightarrow Q] \hookrightarrow \Sigma$ *and* Σ *are sound,*
   then $\Gamma,\Gamma'\,;\Delta,\Delta'\,;\Omega \Longrightarrow \cdot\,;Q$.
4. *If* $\mathsf{act}_\Uparrow(\Gamma\,;\Delta\,;\Omega \Longrightarrow \xi)[\Gamma'\,;\Delta' \Longrightarrow\!\!\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Sigma$ *and* Σ *are sound,*
   then $\Gamma,\Gamma'\,;\Delta,\Delta'\,;\Omega \Longrightarrow R\,;\cdot$ *or* $\Gamma,\Gamma'\,;\Delta,\Delta'\,;\Omega \Longrightarrow \cdot\,;Q$ *depending on whether* $\xi = R$ *or* $\xi = Q$.
5. *If* $\Gamma\,;\Delta \Longrightarrow\!\!\!\!\!\!\Rightarrow Q$, *then* $\Gamma\,;\Delta \Longrightarrow \cdot\,;Q$.

Proof sketch. The first three parts are proven by structural induction on the given rule relations $\mathsf{foc}^+_\Uparrow$, $\mathsf{foc}^-_\Uparrow$ and $\mathsf{act}_\Uparrow$, where the induction hypothesis may be used whenever the focused formula or the height the $\mathsf{act}_\Uparrow$ derivation is smaller. Part 4 is a direct consequence of parts 1 and 2. □

Theorem 6.15.

1. *If* $\Gamma\,;\Delta \gg A$, *then for some* Σ,
   (a) $\mathsf{foc}^+_\Uparrow(A)[\Gamma\,;\Delta \Longrightarrow\!\!\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Sigma$, *and*
   (b) Σ *are all derivable.*
2. *If* $\Gamma\,;\Delta\,;A \ll Q$, *then for some* Σ,
   (a) $\mathsf{foc}^-_\Uparrow(A)[\Gamma\,;\Delta \Longrightarrow\!\!\!\!\!\!\Rightarrow Q] \hookrightarrow \Sigma$, *and*
   (b) Σ *are all derivable.*
3. *If* $\Gamma_1,\Gamma_2\,;\Delta_1,\Delta_2\,;\Omega \Longrightarrow \xi \uplus \gamma$ *(where* $\xi \uplus \gamma$ *means either* ξ *or* γ *is empty), then for some* Σ
   (a) $\mathsf{act}_\Uparrow(\Gamma_1\,;\Delta_1\,;\Omega \Longrightarrow \xi)[\Gamma_2\,;\Delta_2 \Longrightarrow\!\!\!\!\!\!\Rightarrow \gamma] \hookrightarrow \Sigma$,
   (b) Σ *are all derivable.*

Proof. By induction on the structure of the given backward focusing derivation, $\mathcal{D}$. The following are sketches of a few representative cases.

Case $\otimes R$:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma\,;\Delta_1 \gg A \quad \mathcal{D}_2 :: \Gamma\,;\Delta_2 \gg B}{\Gamma\,;\Delta_1,\Delta_2 \gg A \otimes B}\ \otimes R$$

$\mathsf{foc}^+_\Uparrow(A)[\Gamma\,;\Delta_1 \Longrightarrow\!\!\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Sigma_1$ and Σ_1 are derivable — i.h. on $\mathcal{D}_1$

$\mathsf{foc}^+_\Uparrow(B)[\Gamma\,;\Delta_2 \Longrightarrow\!\!\!\!\!\!\Rightarrow \cdot] \hookrightarrow \Sigma_2$ and Σ_2 are derivable — i.h. on $\mathcal{D}_2$

Then note that $\mathsf{foc}^+_\Uparrow(A \otimes B)[\Gamma\,; \Delta_1, \Delta_2 \Longrightarrow\!\!\!\!\!\rightarrow \cdot] \hookrightarrow \Sigma_1 \cdot \Sigma_2$ by the definition of $\mathsf{foc}^+_\Uparrow$.

Case $\&L_1$:

$$\mathcal{D} = \frac{\mathcal{D}' :: \Gamma\,; \Delta\,; A \ll Q}{\Gamma\,; \Delta\,; A \,\&\, B \ll Q}\ \&L_1$$

$\mathsf{foc}^-_\Uparrow(A)[\Gamma\,; \Delta \Longrightarrow\!\!\!\!\!\rightarrow Q] \hookrightarrow \Sigma$ and Σ are derivable　　　　i.h. on $\mathcal{D}'$

Then note that $\mathsf{foc}^-_\Uparrow(A \,\&\, B)[\Gamma\,; \Delta \Longrightarrow\!\!\!\!\!\rightarrow Q] \hookrightarrow \Sigma$ by definition of $\mathsf{foc}^-_\Uparrow$.

Case Case $\otimes L$:

$$\mathcal{D} = \frac{\mathcal{D}' :: \Gamma, \Gamma'\,; \Delta, \Delta'\,; \Omega \cdot A \cdot B \cdot \Omega' \Longrightarrow \gamma}{\Gamma_1, \Gamma_2\,; \Delta_1, \Delta_2\,; \Omega \cdot A \otimes B \cdot \Omega' \Longrightarrow \gamma}\ \otimes L$$

Let γ_1 and γ_2 be such that $\gamma_1 \uplus \gamma_2 = \gamma$.

$\mathsf{act}_\Uparrow(\Gamma_1\,; \Delta_1\,; \Omega \cdot A \cdot B \cdot \Omega' \Longrightarrow \gamma_1)[\Gamma_2\,; \Delta_2 \Longrightarrow\!\!\!\!\!\rightarrow \gamma_2] \hookrightarrow \Sigma$ and Σ are derivable　　i.h.

Then note that $\mathsf{act}_\Uparrow(\Gamma_1\,; \Delta_1\,; \Omega \cdot A \otimes B \cdot \Omega' \Longrightarrow \gamma_1)[\Gamma_2\,; \Delta_2 \Longrightarrow\!\!\!\!\!\rightarrow \gamma_2] \hookrightarrow \Sigma$ by the definition of $\mathsf{act}_\Uparrow$.

Case $\&R$:

$$\mathcal{D} = \frac{\mathcal{D}_1 :: \Gamma_1, \Gamma_2\,; \Omega \Longrightarrow A\,; \cdot \quad \mathcal{D}_2 :: \Gamma_2, \Gamma_2\,; \Omega \Longrightarrow B\,; \cdot}{\Gamma_1, \Gamma_2\,; \Delta_1, \Delta_2\,; \Omega \Longrightarrow A \,\&\, B\,; \cdot}\ \&R$$

$\mathsf{act}_\Uparrow(\Gamma_1\,; \Delta_1\,; \Omega \Longrightarrow A)[\Gamma_2\,; \Delta_2 \Longrightarrow\!\!\!\!\!\rightarrow \cdot] \hookrightarrow \Sigma_1$ and Σ_1 are derivable　　i.h. on $\mathcal{D}_1$

$\mathsf{act}_\Uparrow(\Gamma_1\,; \Delta_1\,; \Omega \Longrightarrow B)[\Gamma_2\,; \Delta_2 \Longrightarrow\!\!\!\!\!\rightarrow \cdot] \hookrightarrow \Sigma_2$ and Σ_2 are derivable　　i.h. on $\mathcal{D}_2$

Then note that $\mathsf{act}_\Uparrow((\Gamma_1\,; \Delta_1\,; \Omega \Longrightarrow A \,\&\, B)[\Gamma_2\,; \Delta_2 \Longrightarrow\!\!\!\!\!\rightarrow \cdot] \hookrightarrow \Sigma_1 \cdot \Sigma_2$ by the definition of $\mathsf{act}_\Uparrow$. □

Corollary 6.16 (Completeness). *If* $\Gamma\,; \Delta\,; \cdot \Longrightarrow \cdot\,; Q$ *then* $\Gamma\,; \Delta \Longrightarrow\!\!\!\!\!\rightarrow Q$.

Proof sketch. Straightforward application of theorem 6.15. The last rule used to derive $\Gamma\,; \Delta\,; \cdot \Longrightarrow \cdot\,; Q$ is one of lfoc, rfoc or copy; correspondingly we have the derived rules $\mathsf{foc}^+_\Uparrow$, $\mathsf{foc}^-_\Uparrow$ and $!\,\mathsf{foc}^-_\Uparrow$. □

6.2.2 Forward derived rules

The essential idea of adapting backward derived rules to the forward direction is fairly simple: instead of producing new goals from a given conclusion, assemble the conclusion from a collection of given premisses. While the approach may seem straightforward, there is however a fairly major difference: the forward direction does not have every proof that is possible in the backward direction. As demonstrated in thm. 3.11, the forward calculus finds stronger proofs; in particular, not all backward focusing proofs have a corresponding forward proof because the premisses of a derived rule might not be matched in the forward direction. Consider, for instance, showing that $\cdot\,; (p \otimes q) \,\&\, r \Longrightarrow\!\!\!\!\!\!\to \top \otimes \top$: there is no forward proof corresponding to the the backward proof that begins by focusing on $(p \otimes q) \,\&\, r$ on the left, because there is no forward sequent that will match the premiss of the $\&L_1$ rule. The completeness theorem in the forward direction is thus a fairly complex result.

We have to slightly generalize the context composition operators into a language of context expressions. In the simplest case, we merely have to add a given proposition to the linear context, irrespective of the weak flag. This happens, for instance, in the "lfoc" rule where the focused proposition is transferred to the linear context. We write this adjunction as usual using a comma. In the more general case, however, we have to combine two context expressions additively or multiplicatively depending on the kind of rule they were involved in; for these uses, we appropriate the same syntax we used for the single step compositions in the previous section.

$$\text{(context expressions)} \qquad \mathcal{D} \quad ::= \quad [\Delta]_w \mid \mathcal{D}, A \mid \mathcal{D}_1 + \mathcal{D}_2 \mid \mathcal{D}_1 \times \mathcal{D}_2$$

Context expressions can be *simplified* into forward contexts in a bottom-up procedure. We write $\mathcal{D} \hookrightarrow [\Delta]_w$ to denote that $\mathcal{D}$ simplifies into $[\Delta]_w$; it has the following rules.

$$\frac{}{[\Delta]_w \hookrightarrow [\Delta]_w} \qquad \frac{\mathcal{D} \hookrightarrow [\Delta]_w}{\mathcal{D}, A \hookrightarrow [\Delta, A]_w} \qquad \frac{\mathcal{D}_1 \hookrightarrow [\Delta_1]_{w_1} \quad \mathcal{D}_2 \hookrightarrow [\Delta_2]_{w_2}}{\mathcal{D}_1 + \mathcal{D}_2 \hookrightarrow [\Delta_1]_{w_1} + [\Delta_2]_{w_2}}$$

$$\frac{\mathcal{D}_1 \hookrightarrow [\Delta_1]_{w_1} \quad \mathcal{D}_2 \hookrightarrow [\Delta_2]_{w_2}}{\mathcal{D}_1 \times \mathcal{D}_2 \hookrightarrow [\Delta_1]_{w_1} \times [\Delta_2]_{w_2}}$$

We proceed to constructing the forward versions of the relations in the earlier section, $\mathsf{foc}^+_\Downarrow$, $\mathsf{foc}^-_\Downarrow$ and $\mathsf{act}_\Downarrow$. These relations take a sequence of forward sequents as input, corresponding to the premisses of the derived rule, and construct the conclusion as their output. Like with the backward derived rules calculus, we use a different sequent arrow, $\longrightarrow\!\!\!\!\!\to$, to distinguish the forward derived rules. As usual, this is a calculus of neutral sequents, i.e., all propositions in Δ (resp. γ) in $\Gamma\,; [\Delta]_w \longrightarrow\!\!\!\!\!\to \gamma$ are left- (resp. right-) synchronous.

The derived rule for positive subformulas is:

$$\frac{s_1 \quad s_2 \quad \cdots \quad s_n \quad \left(\mathsf{foc}^+_{\Downarrow}(Q)[s_1 \cdot s_2 \cdots s_n] \hookrightarrow \Gamma\,;\mathcal{D} \twoheadrightarrow \cdot\right)}{\Gamma\,;\mathcal{D} \twoheadrightarrow Q}\ \mathsf{foc}^+_{\Downarrow}$$

Similarly, for negative propositions, we have two rules:

$$\frac{s_1 \quad s_2 \quad \cdots \quad s_n \quad \left(\mathsf{foc}^-_{\Downarrow}(P)[s_1 \cdot s_2 \cdots s_n] \hookrightarrow \Gamma\,;\mathcal{D} \twoheadrightarrow Q\right)}{\Gamma\,;\mathcal{D}, P \twoheadrightarrow Q}\ \mathsf{foc}^-_{\Downarrow}$$

$$\frac{s_1 \quad s_2 \quad \cdots \quad s_n \quad \left(\mathsf{foc}^-_{\Downarrow}(A)[s_1 \cdot s_2 \cdots s_n] \hookrightarrow \Gamma\,;\mathcal{D} \twoheadrightarrow Q\right)}{\Gamma, A\,;\mathcal{D} \twoheadrightarrow Q}\ !\,\mathsf{foc}^-_{\Downarrow}$$

These sequents with unsimplified contexts are simplified in one step using the following rule:

$$\frac{\Gamma\,;\mathcal{D} \twoheadrightarrow \gamma \quad \mathcal{D} \hookrightarrow [\Delta]_w}{\Gamma\,;[\Delta]_w \twoheadrightarrow \gamma}\ \text{simplify}$$

These relations are defined in fig. 6.4 and 6.5. For the "match" rule, the notation $\gamma\backslash\xi$ is defined as γ if $\xi = \cdot$, and as $\cdot$ if $\gamma = \xi = Q$.

As as simple example, consider the negative subformula $P = p \,\&\, q \multimap r \,\&\, (s \otimes t)$ for which we attempt to match the three sequents $s_1 = \Gamma_1\,;[\Delta_1]_1 \twoheadrightarrow p$, $s_2 = \Gamma_2\,;[\Delta_2]_0 \twoheadrightarrow q$, and $s_3 = \Gamma_3\,;[\Delta_3, s]_1 \twoheadrightarrow \gamma$ with $t \notin \Delta_3$ and $\Delta_1 \subseteq \Delta_2$. Consider the derivation in figure 6.6.

Thus, the application of the full derived rule for P matched against the sequents s_1, s_2 and s_3 is, precisely,

$$\frac{\Gamma_1\,;[\Delta_1]_1 \twoheadrightarrow p \quad \Gamma_2\,;[\Delta_2]_0 \twoheadrightarrow q \quad \Gamma_3\,;[\Delta_3, s]_1 \twoheadrightarrow \gamma}{\Gamma_3, \Gamma_1, \Gamma_2\,;\Big([\Delta_3]_1 \times \big([\Delta_1]_0 + [\Delta_2]_1\big)\Big), P \twoheadrightarrow^1 \gamma}$$

To show soundness, we simply follow the structure of the definitions of $\mathsf{act}_{\Downarrow}$, $\mathsf{foc}^+_{\Downarrow}$ and $\mathsf{foc}^-_{\Downarrow}$. The structure of the proof is similar to those of theorems 3.9 and 5.9. As usual, for the induction to hold we need to generalize the induction hypothesis to state that every weakened form of a weak sequent is sound.

Definition 6.17.

1. *The sequent* $\Gamma\,;[\Delta]_0 \twoheadrightarrow C$ *is* sound *if* $\Gamma\,;\Delta \Longrightarrow C$*.*

right-focal

$$\frac{p \text{ left-biased}}{\mathsf{foc}^+_{\Downarrow}(p)[\cdot] \hookrightarrow \cdot\,;[p]_0 \twoheadrightarrow \cdot}\ \text{linit}$$

$$\frac{\begin{array}{c}\mathsf{foc}^+_{\Downarrow}(A)[\Sigma_1] \hookrightarrow \Gamma_1\,;\mathcal{D}_1 \twoheadrightarrow \cdot\\ \mathsf{foc}^+_{\Downarrow}(A)[\Sigma_2] \hookrightarrow \Gamma_2\,;\mathcal{D}_2 \twoheadrightarrow \cdot\end{array}}{\mathsf{foc}^+_{\Downarrow}(A \otimes B)[\Sigma_1 \cdot \Sigma_2] \hookrightarrow \Gamma_1, \Gamma_2\,;\mathcal{D}_1 \times \mathcal{D}_2 \twoheadrightarrow \cdot}\ \otimes F \qquad \frac{}{\mathsf{foc}^+_{\Downarrow}(\mathbf{1})[\cdot] \hookrightarrow \cdot\,;[\cdot]_0 \twoheadrightarrow \cdot}\ \mathbf{1}F$$

$$\frac{\mathsf{foc}^+_{\Downarrow}(A_i)[\Sigma] \hookrightarrow s}{\mathsf{foc}^+_{\Downarrow}(A_1 \oplus A_2)[\Sigma] \hookrightarrow s}\ \oplus F_i \qquad \frac{\mathsf{act}_{\Downarrow}(\cdot\,;\cdot\,;\cdot \Longrightarrow A)[\Sigma] \hookrightarrow \Gamma\,;[\cdot]_w \twoheadrightarrow \cdot}{\mathsf{foc}^+_{\Downarrow}(!\,A)[\Sigma] \hookrightarrow \Gamma\,;[\cdot]_0 \twoheadrightarrow \cdot}\ !F$$

$$\frac{\mathsf{foc}^+_{\Downarrow}([t/x]A)[\Sigma] \hookrightarrow s}{\mathsf{foc}^+_{\Downarrow}(\exists x. A)[\Sigma] \hookrightarrow s}\ \exists F \qquad \frac{\mathsf{act}_{\Downarrow}(\cdot\,;\cdot\,;\cdot \Longrightarrow R)[\Sigma] \hookrightarrow s}{\mathsf{foc}^+_{\Downarrow}(R)[\Sigma] \hookrightarrow s}\ FA^+$$

left-focal

$$\frac{p \text{ right-biased}}{\mathsf{foc}^-_{\Downarrow}(p)[\cdot] \hookrightarrow \cdot\,;[\cdot]_0 \twoheadrightarrow p}\ \text{rinit}$$

$$\frac{\mathsf{foc}^-_{\Downarrow}(A_i)[\Sigma] \hookrightarrow s}{\mathsf{foc}^-_{\Downarrow}(A_1 \,\&\, A_2)[\Sigma] \hookrightarrow s}\ \& F_i \qquad \frac{\begin{array}{c}\mathsf{foc}^-_{\Downarrow}(B)[\Sigma_1] \hookrightarrow \Gamma_1\,;\mathcal{D}_1 \twoheadrightarrow \gamma\\ \mathsf{foc}^+_{\Downarrow}(A)[\Sigma_2] \hookrightarrow \Gamma_2\,;\mathcal{D}_2 \twoheadrightarrow \cdot\end{array}}{\mathsf{foc}^-_{\Downarrow}(A \multimap B)[\Sigma_1 \cdot \Sigma_2] \hookrightarrow \Gamma_1, \Gamma_2\,;\mathcal{D}_1 \times \mathcal{D}_2 \twoheadrightarrow \gamma}\ \multimap F$$

$$\frac{\mathsf{foc}^-_{\Downarrow}([t/x]A)[\Sigma] \hookrightarrow s}{\mathsf{foc}^-_{\Downarrow}(\forall x. A)[\Sigma] \hookrightarrow s}\ \forall F \qquad \frac{\mathsf{act}_{\Downarrow}(\cdot\,;\cdot\,;L \Longrightarrow \cdot)[\Sigma] \hookrightarrow s}{\mathsf{foc}^-_{\Downarrow}(L)[\Sigma] \hookrightarrow s}\ FA^-$$

Figure 6.4: Forward derived rules: focal phase

2. *The sequent* $\Gamma\,;[\Delta]_1 \twoheadrightarrow \gamma$ *is* sound *if* $\Gamma\,;\Delta' \Longrightarrow C$ *for every* $\Delta' \supseteq \Delta$ *and* $C \supseteq \gamma$.

Lemma 6.18. *If* Σ *are sound, then*

1. *If* $\mathsf{foc}^+_{\Downarrow}(A)[\Sigma] \hookrightarrow \Gamma\,;[\Delta]_w \twoheadrightarrow \cdot$, *then* $\Gamma\,;[\Delta]_w \twoheadrightarrow A$ *is sound.*
2. *If* $\mathsf{foc}^-_{\Downarrow}(A)[\Sigma] \hookrightarrow \Gamma\,;[\Delta]_w \twoheadrightarrow \gamma$, *then* $\Gamma\,;[\Delta, A]_w \twoheadrightarrow \gamma$ *is sound.*
3. *If* $\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow \gamma)[\Sigma] \hookrightarrow \Gamma'\,;[\Delta']_w \twoheadrightarrow \gamma'$, *then*
   (a) *if* $w = 0$, *then* $\gamma = C$ *and* $\Gamma, \Gamma'\,;\Delta, \Delta'\,;\Omega \Longrightarrow C\,;\cdot$.
   (b) *if* $w = 1$, *then* $\Gamma, \Gamma'\,;\Delta''\,;\Omega \Longrightarrow C\,;\cdot$ *for any* $\Delta'' \supseteq \Delta, \Delta'$ *and* $C \supseteq \gamma$.

Proof sketch. Structural induction the definitions of $\mathsf{foc}^+_{\Downarrow}$, $\mathsf{foc}^-_{\Downarrow}$ and $\mathsf{act}_{\Downarrow}$. The proof in the forward direction is essentially identical to that of theorem 6.14. □

Corollary 6.19 (soundness)**.** *If* $\Gamma\,;[\Delta]_w \twoheadrightarrow \gamma$ *is derivable, then it is sound.*

active (ξ is of the form $\cdot$ or Q)

$$\frac{\begin{array}{c}\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow A)[\Sigma_1] \hookrightarrow \Gamma_1\,;\mathcal{D}_1 \twoheadrightarrow \cdot\\ \mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow B)[\Sigma_2] \hookrightarrow \Gamma_2\,;\mathcal{D}_2 \twoheadrightarrow \cdot\end{array}}{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow A \,\&\, B)[\Sigma_1 \cdot \Sigma_2] \hookrightarrow \Gamma_1, \Gamma_2\,;\mathcal{D}_1 + \mathcal{D}_2 \twoheadrightarrow \cdot}\ \&A$$

$$\frac{}{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow \top)[\cdot] \hookrightarrow \cdot\,;[\cdot]_1 \twoheadrightarrow \cdot}\ \top A$$

$$\frac{\begin{array}{c}\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot A \cdot \Omega' \Longrightarrow \xi)[\Sigma_1] \hookrightarrow \Gamma_1\,;\mathcal{D}_1 \twoheadrightarrow \gamma_1\\ \mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot B \cdot \Omega' \Longrightarrow \xi)[\Sigma_2] \hookrightarrow \Gamma_2\,;\mathcal{D}_2 \twoheadrightarrow \gamma_2\end{array}}{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot A \oplus B \cdot \Omega' \Longrightarrow \xi)[\Sigma_1 \cdot \Sigma_2] \hookrightarrow \Gamma_1 \cup \Gamma_2\,;\mathcal{D}_1 + \mathcal{D}_2 \twoheadrightarrow \gamma_1 \cup \gamma_2}\ \oplus A$$

$$\frac{}{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot \mathbf{0} \cdot \Omega' \Longrightarrow \xi)[\cdot] \hookrightarrow \cdot\,;[\cdot]_1 \twoheadrightarrow \cdot}\ \mathbf{0}A$$

$$\frac{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot A \Longrightarrow B)[\Sigma] \hookrightarrow s}{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow A \multimap B)[\Sigma] \hookrightarrow s}\ \multimap A$$

$$\frac{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow B)[\Sigma] \hookrightarrow \Gamma'\,;[\Delta']_1 \twoheadrightarrow \cdot}{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow A \multimap B)[\Sigma] \hookrightarrow \Gamma'\,;[\Delta']_1 \twoheadrightarrow \cdot}\ \multimap A'$$

$$\frac{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot A \cdot B \cdot \Omega' \Longrightarrow \xi)[\Sigma] \hookrightarrow s}{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot A \otimes B \cdot \Omega' \Longrightarrow \xi)[\Sigma] \hookrightarrow s}\ \otimes A$$

$$\frac{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot A_i \cdot \Omega' \Longrightarrow \xi)[\Sigma] \hookrightarrow \Gamma'\,;[\Delta']_1 \twoheadrightarrow \gamma}{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot A_1 \otimes A_2 \cdot \Omega' \Longrightarrow \xi)[\Sigma] \hookrightarrow \Gamma'\,;[\Delta']_1 \twoheadrightarrow \gamma}\ \otimes A'$$

$$\frac{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow \xi)[\Sigma] \hookrightarrow \Gamma'\,;[\Delta']_w \twoheadrightarrow \xi'}{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot \mathbf{1} \Longrightarrow \xi)[\Sigma] \hookrightarrow \Gamma'\,;[\Delta']_w \twoheadrightarrow \xi'}\ \mathbf{1}A$$

$$\frac{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot \Omega' \Longrightarrow \xi)[\Sigma] \hookrightarrow s}{\mathsf{act}_{\Downarrow}(\Gamma \backslash A\,;\Delta\,;\Omega \cdot {!}A \cdot \Omega' \Longrightarrow \xi)[\Sigma] \hookrightarrow s}\ {!}A$$

$$\frac{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow [a/x]A)[\Sigma] \hookrightarrow s}{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \Longrightarrow \forall x.\, A)[\Sigma] \hookrightarrow s}\ \forall A^a$$

$$\frac{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot [a/x]A \cdot \Omega' \Longrightarrow \xi)[\Sigma] \hookrightarrow s}{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot \exists x.\, A \cdot \Omega' \Longrightarrow \xi)[\Sigma] \hookrightarrow s}\ \exists A^a$$

$$\frac{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta, P\,;\Omega \cdot \Omega' \Longrightarrow \xi)[\Sigma] \hookrightarrow s}{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\Omega \cdot P \cdot \Omega' \Longrightarrow \xi)[\Sigma] \hookrightarrow s}\ \mathsf{act}$$

$$\frac{\xi \subseteq \gamma}{\mathsf{act}_{\Downarrow}(\Gamma\,;\Delta\,;\cdot \Longrightarrow \xi)[\Gamma, \Gamma'\,;[\Delta, \Delta']_w \twoheadrightarrow \gamma] \hookrightarrow \Gamma'\,;[\Delta']_w \twoheadrightarrow \gamma \backslash \xi}\ \text{match}$$

Figure 6.5: forward derived rules: active phase

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{\mathsf{act}_{\Downarrow}(\cdot\,;s\,;\cdot \Longrightarrow \cdot)[s_3] \hookrightarrow \Gamma_3\,;[\Delta_3]_1 \twoheadrightarrow \gamma}
}{\mathsf{act}_{\Downarrow}(\cdot\,;\cdot\,;s \Longrightarrow \cdot)[s_3] \hookrightarrow \Gamma_3\,;[\Delta_3]_1 \twoheadrightarrow \gamma}
}{\mathsf{act}_{\Downarrow}(\cdot\,;\cdot\,;s \otimes t \Longrightarrow \cdot)[s_3] \hookrightarrow \Gamma_3\,;[\Delta_3]_1 \twoheadrightarrow \gamma}\otimes A'
}{\mathsf{foc}^{-}_{\Downarrow}(s \otimes t)[s_3] \hookrightarrow \Gamma_3\,;[\Delta_3]_1 \twoheadrightarrow \gamma}
}{\mathsf{foc}^{-}_{\Downarrow}(r \mathbin{\&} (s \otimes t))[s_3] \hookrightarrow \Gamma_3\,;[\Delta_3]_1 \twoheadrightarrow \gamma}
\quad
\dfrac{
\dfrac{
\dfrac{}{\mathsf{act}_{\Downarrow}(\cdot\,;\cdot\,;\cdot \Longrightarrow p)[s_1] \hookrightarrow \Gamma_1\,;[\Delta_1]_1 \twoheadrightarrow \cdot}
}{\mathsf{foc}^{+}_{\Downarrow}(p)[s_1] \hookrightarrow \Gamma_1\,;[\Delta_1]_1 \twoheadrightarrow \cdot}
\quad
\dfrac{
\dfrac{}{\mathsf{act}_{\Downarrow}(\cdot\,;\cdot\,;\cdot \Longrightarrow q)[s_2] \hookrightarrow \Gamma_2\,;[\Delta_2]_0 \twoheadrightarrow \cdot}
}{\mathsf{foc}^{+}_{\Downarrow}(q)[s_2] \hookrightarrow \Gamma_2\,;[\Delta_2]_0 \twoheadrightarrow \cdot}
}{\mathsf{foc}^{+}_{\Downarrow}(p \mathbin{\&} q)[s_1 \cdot s_2] \hookrightarrow \Gamma_1, \Gamma_2\,;[\Delta_1]_1 + [\Delta_2]_0 \twoheadrightarrow \cdot}
}{\mathsf{foc}^{-}_{\Downarrow}(P)[s_3 \cdot s_1 \cdot s_2] \hookrightarrow \Gamma_3, \Gamma_1, \Gamma_2\,;[\Delta_3]_1 \times \big([\Delta_1]_1 + [\Delta_2]_0\big) \twoheadrightarrow^{1} \gamma}
$$

Figure 6.6: An example forward derived rule

Proof sketch. The last rule used to derive $\Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma$ is one of $\mathsf{foc}^+_\Downarrow$, $\mathsf{foc}^-_\Downarrow$ or $!\,\mathsf{foc}^-_\Downarrow$. Each case is a direct application of lemma 6.18. □

Next we show that the forward derived rule calculus is complete with respect to the backward derived rule calculus of sec. 6.2.1.

Definition 6.20. *For any context* Γ*, the context* $|\Gamma|$ *stands for the maximally contracted form* Γ*.*

Definition 6.21 (Stronger forms)**.**

1. *A forward derived sequent* $\Gamma\,;[\mathcal{D}]_w \longrightarrow\!\!\!\!\rightarrow \gamma$ *is said to be stronger than a backward derived sequent* $\Gamma'\,;\Delta' \Longrightarrow\!\!\!\!\!\rightarrow \gamma'$*, written as the relation* $\preceq$*, if* $|\Gamma| \subseteq \Gamma'$ *and*
   (a) *if* $w = 0$ *then* $\Delta = \Delta'$ *and* $\gamma = \gamma'$*; and*
   (b) *if* $w = 1$ *then* $\Delta \subseteq \Delta'$ *and* $\gamma \subseteq \gamma'$*.*
2. *A forward unsimplified sequent* $\Gamma\,;\mathcal{D} \longrightarrow\!\!\!\!\rightarrow \gamma$ *is said to be stronger than a backward derived sequent* $\Gamma'\,;\Delta' \Longrightarrow\!\!\!\!\!\rightarrow \gamma'$ *if for every* $[\Delta]_w$ *such that* $\mathcal{D} \hookrightarrow [\Delta]_w$*, the sequent* $\Gamma\,;[\Delta]_w \longrightarrow\!\!\!\!\rightarrow \gamma$ *is stronger than* $\Gamma'\,;\Delta' \Longrightarrow\!\!\!\!\!\rightarrow \gamma'$*.*

Lemma 6.22.

1. *If* $\mathsf{foc}^+_\Uparrow(A)[s] \hookrightarrow \Sigma$ *and there exists a derivable sequence of sequents,* $\Sigma' \preceq \Sigma$ *for which* $\mathsf{foc}^+_\Downarrow(A)[\Sigma'] \hookrightarrow s'$*, then* $s' \preceq s$*.*
2. *If* $\mathsf{foc}^-_\Uparrow(A)[s] \hookrightarrow \Sigma$ *and there exists a derivable sequence* $\Sigma' \preceq \Sigma$ *for which* $\mathsf{foc}^-_\Downarrow(A)[\Sigma'] \hookrightarrow s'$*, then* $s' \preceq s$*.*
3. *If* $\mathsf{act}_\Uparrow(\Gamma\,;\Delta\,;\Omega \Longrightarrow \xi)[s] \hookrightarrow \Sigma$ *and there exists a derivable sequence* $\Sigma' \preceq \Sigma$*, and* $\Gamma' \subseteq \Gamma$*,* $\Delta' \subseteq \Delta$*,* $\Omega' \subseteq \Omega$ *and* $\xi' \subseteq \xi$ *for which* $\mathsf{act}_\Downarrow(\Gamma'\,;\Delta'\,;\Omega' \Longrightarrow \xi')[\Sigma'] \hookrightarrow s'$*, then* $s' \preceq s$*.*

Proof sketch. By induction on the structure of the $\mathsf{foc}^-_\Uparrow$, $\mathsf{foc}^+_\Uparrow$ and $\mathsf{act}_\Uparrow$ derivations. The following is a representative case for $\otimes F$.

$$\frac{\mathsf{foc}^+_\Uparrow(A)[\Gamma\,;\Delta_1 \Longrightarrow\!\!\!\!\!\rightarrow \cdot] \hookrightarrow \Sigma_1 \qquad \mathsf{foc}^+_\Uparrow(A)[\Gamma\,;\Delta_2 \Longrightarrow\!\!\!\!\!\rightarrow \cdot] \hookrightarrow \Sigma_2}{\mathsf{foc}^+_\Uparrow(A \otimes B)[\Gamma\,;\Delta_1, \Delta_2 \Longrightarrow\!\!\!\!\!\rightarrow \cdot] \hookrightarrow \Sigma_1 \cdot \Sigma_2}\ \otimes F$$

Suppose there is a derivable $(\Sigma_1' \cdot \Sigma_2') \prec (\Sigma_1 \cdot \Sigma_2)$ such that

$$\mathsf{foc}^+_\Downarrow(A \otimes B)[\Sigma_1' \cdot \Sigma_2'] \hookrightarrow \Gamma_1, \Gamma_2\,;\mathcal{D}_1 \times \mathcal{D}_2 \longrightarrow\!\!\!\!\rightarrow \cdot$$

where $\mathsf{foc}^+_\Downarrow(A)[\Sigma'_1] \hookrightarrow \Gamma_2 \,;\, \mathcal{D}_2 \longrightarrow\!\!\!\!\!\rightarrow \cdot$ and $\mathsf{foc}^+_\Downarrow(S)[\Sigma'_2] \hookrightarrow \Gamma_2 \,;\, \mathcal{D}_2 \longrightarrow\!\!\!\!\!\rightarrow \cdot$. By the induction hypotheses $|\Gamma_1| \subseteq \Gamma$ and $|\Gamma_2| \subseteq \Gamma$. Therefore $|\Gamma_1, \Gamma_2| \subseteq \Gamma$. Next, let $[\Delta_1]_{w_1}$ and $[\Delta_2]_{w_2}$ be given such that $\mathcal{D}_1 \hookrightarrow [\Delta'_1]_{w_1}$ and $\mathcal{D}_2 \hookrightarrow [\Delta'_2]_{w_2}$. If $w_1 = w_2 = 0$, then $\Delta'_1 = \Delta_1$ and $\Delta'_2 = \Delta_2$, so $[\Delta'_1]_{w_1} + [\Delta'_2]_{w_2} = \Delta_1, \Delta_2$. If $w_2 = 1$, then $\Delta'_2 \subseteq \Delta_2$, so $[\Delta'_1]_{w_1} + [\Delta'_2]_{w_2} \subseteq \Delta_1, \Delta_2$. The case of $w_1 = 1$ is similar. □

Corollary 6.23 (completeness). *If* $\Gamma \,;\, \Delta \Longrightarrow\!\!\!\!\!\Rightarrow Q$, *then for some derivable* $\Gamma' \,;\, [\Delta']_w \longrightarrow\!\!\!\!\!\rightarrow \gamma$,

$$(\Gamma' \,;\, [\Delta']_w \longrightarrow\!\!\!\!\!\rightarrow \gamma) \leq (\Gamma \,;\, \Delta \Longrightarrow\!\!\!\!\!\Rightarrow Q).$$

Proof sketch. By induction on the derivation of $\Gamma \,;\, \Delta \Longrightarrow\!\!\!\!\!\Rightarrow Q$. There are three cases—one each for $\mathsf{foc}^-_\Uparrow$, $!\,\mathsf{foc}^-_\Uparrow$ and $\mathsf{foc}^+_\Uparrow$—used to derive $\Gamma \,;\, \Delta \Longrightarrow\!\!\!\!\!\Rightarrow Q$. In each case, we use lem. 6.22 cases 1 or 2, as appropriate to drive the induction. □

6.3 The focused inverse method

What remains is to implement a search strategy that uses the forward calculus. The primary issue in the forward direction is to enumerate the propositions for which we need to derive inference rules. As the calculus of derived rules has only neutral sequents as premisses and conclusions, we need only generate rules for propositions that occur in neutral sequents; we call them *frontier propositions*. To find the frontier propositions in a goal sequent, we abstractly replay the focusing and active phases to identify the phase transitions. Each transition from an active to a focal phase produces a frontier proposition. Formally, we define two generating functions, f (focal) and a (active), from signed propositions to multisets of frontier propositions. None of the logical constants are in the frontier as we never need to construct explicit rules for them, as the conclusions of rules such as $\top R$ and $\mathbf{1}R$ are easy to predict. Similarly we do not count a negative left-biased focused atom (or a positive right-biased focused atom) in the frontier as these will be derived using an init rule (linit or rinit) for which there are no premisses. The result of this computation will produce a decorated subformula (see defn. 4.4).

$$
\begin{array}{rlrll}
f(p)^- &= \emptyset & f(p)^+ = a(p)^\pm &= \{p^\pm_\cdot\} & \text{if } p \text{ right-biased} \\
f(p)^+ &= \emptyset & f(p)^- = a(p)^\mp &= \{p^\mp_\cdot\} & \text{if } p \text{ left-biased} \\
f(A \otimes B)^- &= a(A \otimes B)^- & f(A \otimes B)^+ &= f(A)^+, f(B)^+ &
\end{array}
$$

$$a(A \otimes B)^- = a(A)^-, a(B)^- \qquad a(A \otimes B)^+ = f(A \otimes B)^+, (A \otimes B)^-_{\cdot}$$

$$f(A \mathbin{\&} B)^- = f(A)^-, f(B)^- \qquad f(A \mathbin{\&} B)^+ = a(A \mathbin{\&} B)^+$$
$$a(A \mathbin{\&} B)^- = f(A \mathbin{\&} B)^-, (A \mathbin{\&} B)^-_{\cdot} \qquad a(A \mathbin{\&} B)^+ = a(A)^+, a(B)^+$$

$$f(A \multimap B)^- = f(A)^+, f(B)^- \qquad f(A \multimap B)^+ = a(A \multimap B)^+$$
$$a(A \multimap B)^- = f(A \multimap B)^-, (A \multimap B)^-_{\cdot} \qquad a(A \multimap B)^+ = a(A)^-, a(B)^+$$

$$f(!A)^- = a(!A)^- \qquad f(!A)^+ = a(A)^+$$
$$a(!A)^- = f(A)^-, (A)^-_{!} \qquad a(!A)^+ = f(A)^+, (!A)^+_{\cdot}$$

$$f(\mathbf{1})^\pm = a(\mathbf{1})^\pm = \emptyset \qquad f(\top)^\pm = a(\top)^\pm = \emptyset$$

For example, $f(p \mathbin{\&} q \multimap r \mathbin{\&} (!s \otimes t))^- = p^+_{\cdot}, q^+_{\cdot}, s^-_{!}, t^-_{\cdot}$.

Definition 6.24 (frontier). *Given a goal $\Gamma ; \Delta \Longrightarrow Q$, its frontier contains:*

i. all (top-level) propositions in Γ, Δ, Q;
ii. for any $A \in \Gamma, \Delta$, the collection $f(A)^-$; and
iii. the collection $f(Q)^+$.

Lemma 6.25 (neutral subformula property). *In any backward focused proof, all neutral sequents consist only of frontier propositions of the goal sequent.*

Proof sketch. By structural induction on the given derivation. We omit the easy details. □

In the preparatory phase for the inverse method, we calculate the frontier propositions of the goal sequent. There is no need to generate initial sequents separately, as the executions of negative atoms in the frontier directly give us the necessary initial sequents.

During the search procedure, each rule is applied to sequents selected from the current database, and if the rule applies successfully then we get a new sequent, which is then considered for insertion in the database. It is possible (and common) that a generated sequent is actually subsumed by some sequent already in the database (forward subsumption). It is also possible (though less common) for a new sequent to be stronger than some sequents already in the database. In this case, the old weaker sequents are no longer considered for new derivations (backward subsumption).

6.3.1 Implementation details

In this section we shall describe briefly the main details concerning the implementation of the inverse method of section 5.6. The first of these concerns the implementation of inference rules. Given a proposition in the frontier, say A^+, the focusing phase of the derived rule for it is completely predictable once we know which disjunctive choices were taken. Once at the active phase, however, the precise match that will be generated will depend on whether the corresponding input sequent is weak or not. The implementation therefore proceeds as follows. For every derived rule, we fix the precise choices that will be taken. Then, for each rule in $\mathsf{foc}^+_\Downarrow$ or $\mathsf{foc}^-_\Downarrow$ relation is interpreted as a *reusable continuation* (sometimes called *handlers*). These continuations expect as arguments the current intermediate state of the focusing phase, and produce as result the output of the rule (if there are no more premisses), or another handler if there are still other branches of the proof tree to explore.

For a binary rule such as $\otimes R$, the handler for the left operand is sequenced with that of the right in the obvious fashion; thus, these handlers force a specific order in which the premisses of the derived rule must be met. Therefore, it is important for the rule application engine to be complete no matter which order the premisses have to be satisfied. The percolation algorithm outlined in sec. 5.6 (defn. 5.30) guarantees this completeness.

The second implementation detail has to do with the $!\,\mathsf{foc}^-_\Downarrow$ rule. If the focused proposition $A^-_!$ actually occurs in the unrestricted context in the final goal sequent $\Gamma_0\,;[\Delta_0]_0 \longrightarrow \gamma_0$, then it doesn't actually need to be inserted into the unrestricted context in the conclusion. The reason for this is that in the backwards calculus the context Γ_0 will be shared in every branch of the proof, so one thinks of it as part of the ambient state of the prover instead of representing it explicitly as part of the current goal. Hence, in the forward direction there is never any need to explicitly record Γ_0 or portions of it in any generated sequent. Thus we obtain two versions of the $!\,\mathsf{foc}^-_\Downarrow$*rule*:

$$\frac{s_1 \quad s_2 \quad \cdots \quad s_n \quad \left(\mathsf{foc}^-_\Downarrow(A)[s_1 \cdot s_2 \cdots s_n] \hookrightarrow \Gamma\,;\mathcal{D} \longrightarrow\!\!\!\!\rightarrow Q\right) \quad A \notin \Gamma_0}{\Gamma, A\,;\mathcal{D} \longrightarrow\!\!\!\!\rightarrow Q}\;!\,\mathsf{foc}^-_\Downarrow$$

and

$$\frac{s_1 \quad s_2 \quad \cdots \quad s_n \quad \left(\mathsf{foc}^{-}_{\Downarrow}(A)[s_1 \cdot s_2 \cdots s_n] \hookrightarrow \Gamma \,;\, \mathcal{D} \longrightarrow\!\!\!\!\rightarrow Q\right) \quad A \in \Gamma_0}{\Gamma \,;\, \mathcal{D} \longrightarrow\!\!\!\!\rightarrow Q}\ !\,\mathsf{foc}^{-}_{\Downarrow}\text{-delete}$$

This optimisation sometimes goes by the name of *globalisation*. In the implementation syntax (described in more detail in chapter 7), all declared propositions are counted as belonging to this global state and globalised. If the final goal proposition is itself asynchronous, then the *bact*+ phase is played out for this proposition to obtain a collection of goal sequents, and any additional propositions entered into the unrestricted context of these goal sequents is also treated as global.

6.4 Embedding non-linear logics

6.4.1 Intuitionistic logic

There have been many proposed embeddings of ordinary (non-linear) logics into linear logic using the exponential operator [42, 27] that translate sub-formulas uniformly. These translations do not preserve the focusing properties of the source logic because the use of the exponential ! operator causes loss of focus, as mentioned in sec. 6.1. For example, $a \vee b \vee c$ never blurs focus at $b \vee c$, but if we use the Girard embedding (defn. 2.5), then the translation $!a \oplus !(!b \oplus !c)$ causes a loss of focus at $!b \oplus !c$. It is possible though to give a focusing-aware translation that is faithful to the focusing system of the source logic. As an example, consider the basic intuitionistic propositional logic with connectives $\{\wedge, \mathtt{t}, \vee, \mathtt{f}, \supset\}$. The focusing system for this logic treats $\wedge$ as both synchronous and asynchronous on both sides. The rules are as follows:

$$\frac{}{\Gamma \,;\, p \ll_I p} \qquad \frac{\Gamma \,;\, A_i \ll_I Q}{\Gamma \,;\, A_1 \wedge A_2 \ll_I Q} \qquad \frac{\Gamma \,;\, B \ll_I Q \quad \Gamma \gg_I A}{\Gamma \,;\, A \supset B \ll_I Q}$$

$$\frac{\Gamma \gg_I A \quad \Gamma \gg_I B}{\Gamma \gg_I A \wedge B} \qquad \frac{}{\Gamma \gg_I \mathtt{t}} \qquad \frac{\Gamma \gg_I A_i}{\Gamma \gg_I A_1 \vee A_2}$$

$$\frac{\Gamma \,;\, \Omega \Longrightarrow_I A \,;\, \cdot \quad \Gamma \,;\, \Omega \Longrightarrow_I B \,;\, \cdot}{\Gamma \,;\, \Omega \Longrightarrow_I A \wedge B \,;\, \cdot} \qquad \frac{}{\Gamma \,;\, \Omega \Longrightarrow_I \mathtt{t} \,;\, \cdot} \qquad \frac{\Gamma \,;\, \Omega \cdot A \Longrightarrow_I B \,;\, \cdot}{\Gamma \,;\, \Omega \Longrightarrow_I A \supset B \,;\, \cdot}$$

$$\frac{\Gamma\,;\Omega\cdot A\cdot B\cdot\Omega'\Longrightarrow_I\gamma}{\Gamma\,;\Omega\cdot A\wedge B\cdot\Omega'\Longrightarrow_I\gamma}\qquad\frac{\Gamma\,;\Omega\cdot A\cdot\Omega'\Longrightarrow_I\gamma\quad\Gamma\,;\Omega\cdot B\cdot\Omega'\Longrightarrow_I\gamma}{\Gamma\,;\Omega\cdot A\vee B\cdot\Omega'\Longrightarrow_I\gamma}\qquad\frac{}{\Gamma\,;\Omega\cdot\mathtt{f}\cdot\Omega'\Longrightarrow_I\gamma}$$

$$\frac{\Gamma,P\,;\Omega\cdot\Omega'\Longrightarrow_I\gamma}{\Gamma\,;\Omega\cdot P\cdot\Omega'\Longrightarrow_I\gamma}\text{act}\qquad\frac{\Gamma\gg_I Q\quad Q\text{ non atomic}}{\Gamma\,;\cdot\Longrightarrow_I\cdot\,;Q}\qquad\frac{\Gamma\,;P\ll_I Q}{\Gamma,P\,;\cdot\Longrightarrow_I\cdot\,;Q}$$

$$\frac{\Gamma\,;\cdot\Longrightarrow_I R\,;\cdot}{\Gamma\gg_I R}\qquad\frac{\Gamma\,;L\Longrightarrow_I\cdot\,;Q}{\Gamma\,;L\ll_I Q}$$

Here, γ is of the form $\cdot\,;Q$ or $C\,;\cdot$. Note that in this formulation atomic propositions are always right-biased to keep things simple. Extending this propositional translation to the first-order setting is also easy.

We intend to translate signed intuitionistic formulas to signed linear formulas in a way that preserves the focusing structure of proofs. The translation is modal with two phases: A (active) and F (focal). A positive focal (and negative active) $\wedge$ is translated as $\otimes$, and the duals as &. For every use of the act rule, the corresponding translation phase affixes an exponential; the phase-transitions in the image of the translation exactly mirror those in the source.

$$\begin{aligned}
F(p)^- &= p & F(p)^+ &= p\\
A(p)^- &= {!}\,p & A(p)^+ &= p\\[1ex]
F(A\wedge B)^- &= F(A)^-\mathbin{\&}F(B)^- & F(A\wedge B)^+ &= F(A)^+\otimes F(B)^+\\
A(A\wedge B)^- &= A(A)^-\otimes A(B)^- & A(A\wedge B)^+ &= A(A)^+\mathbin{\&}A(B)^+\\[1ex]
F(\mathtt{t})^- &= \top & F(\mathtt{t})^+ &= \mathbf{1}\\
A(\mathtt{t})^- &= \mathbf{1} & A(\mathtt{t})^+ &= \top\\[1ex]
F(A\vee B)^- &= A(A\vee B)^- & F(A\vee B)^+ &= F(A)^+\oplus F(B)^+\\
A(A\vee B)^- &= A(A)^-\oplus A(B)^- & A(A\vee B)^+ &= F(A\vee B)^+\\[1ex]
F(\mathtt{f})^- &= \mathbf{0} & F(\mathtt{f})^+ &= \mathbf{0}\\
A(\mathtt{f})^- &= \mathbf{0} & A(\mathtt{f})^+ &= \mathbf{0}\\[1ex]
F(A\supset B)^- &= F(A)^+\multimap F(B)^- & F(A\supset B)^+ &= A(A\supset B)^+\\
A(A\supset B)^- &= {!}\,F(A\supset B)^- & A(A\supset B)^+ &= A(A)^-\multimap A(B)^+
\end{aligned}$$

The reverse translation, written $-^o$, is trivial: simply erase all !s, rewrite & and $\otimes$ as $\wedge$,

$\multimap$ as $\supset$, and $\oplus$ as $\vee$. The faithfulness of the translations can be established as a pair of soundness and completeness theorems, provable by simple structural induction.

Theorem 6.26. *Soundness:*

1. *If* $\Gamma \,;\cdot \gg A$ *then* $\Gamma^o \gg_I A^o$.
2. *If* $\Gamma \,;\cdot\,; A \ll Q$ *then* $\Gamma^o \,; A^o \ll_I Q^o$.
3. *If* $\Gamma \,;\cdot\,; \Omega \Longrightarrow \gamma$ *then* $\Gamma^o \,; \Omega^o \Longrightarrow_I \gamma^o$ *(where* γ *is of the form* $C\,;\cdot$ *or* $\cdot\,;Q$*.)*

Completeness:

1. *If* $\Gamma \gg_I A$ *then* $F(\Gamma)^- \,;\cdot \gg F(A)^+$.
2. *If* $\Gamma \,; A \ll_I Q$ *then* $F(\Gamma)^- \,;\cdot\,; F(A)^- \ll F(Q)^+$.
3. *If* $\Gamma \,; \Omega \Longrightarrow_I \cdot\,; Q$ *then* $F(\Gamma)^- \,;\cdot\,; A(\Omega)^- \Longrightarrow \cdot\,; F(Q)^+$.
4. *If* $\Gamma \,; \Omega \Longrightarrow_I R\,;\cdot$ *then* $F(\Gamma)^- \,;\cdot\,; A(\Omega)^- \Longrightarrow A(R)^+ \,;\cdot$.

Proof sketch. Soundness is immediate because the linear sequent calculus is simply a refinement of the intuitionistic calculus. Completeness is established by straightforward structural induction on the given intuitionistic derivations. We omit the rather easy details. □

An important feature of this translation is that only negative atoms and implications are !-affixed; this mirrors a similar observation by Dyckhoff that the ordinary intuitionistic logic has a contraction-free sequent calculus that only needs to duplicate negative atoms and implications [37]. Dyckhoff's calculus however has no notion of focusing, so this isn't a precise correspondence; incorporating focusing into this calculus is currently an open question.

6.4.2 The Horn fragment

In complex specifications that employ linearity, there are often significant sub-specifications that lie in the Horn fragment. Unfortunately, the basic inverse method is quite inefficient on Horn formulas, as already noted by Tammet [110]. His prover switches between hyperresolution for Horn and near-Horn formulas and the inverse method for other propositions.

With focusing, this *ad hoc* strategy selection becomes entirely unnecessary. The focused inverse method for intuitionistic linear logic, when applied to a classical, non-linear Horn formula, will exactly behave as classical hyperresolution or SLD resolution depending on the focusing bias of the atomic propositions. This remarkable property gives further credence to the power of focusing as a technique for forward reasoning. In the next two sections we will describe this correspondence in slightly more detail.

A Horn clause has the form $\neg p_1, \ldots, \neg p_n, p$ where the p_i and p are atomic predicates over their free variables. This can easily be generalized to include conjunction and truth, but we restrict our attention to this simple clausal form, as theories with conjunction and truth can be simplified into this form. A Horn theory Ψ is just a set of Horn clauses, and a Horn query is of the form $\Psi \vdash g$ where g is a ground atomic "goal" formula[4]. In the following section we use a simple translation $(-)^o$ of these Horn clauses into linear logic where $\neg p_1, \ldots, \neg p_n, p$ containing the free variables $\vec{x}$ is translated into $\forall \vec{x}. p_1 \multimap \cdots \multimap p_n \multimap p$, and the query $\Psi \vdash g$ is translated as $(\Psi)^o ; [\cdot]_0 \longrightarrow g$. This is a special case of a general focus-preserving translation of sec. 6.4.1.

6.4.3 Hyperresolution

The hyperresolution strategy for the Horn query $\Psi \vdash g$ is just forward reasoning with the following rule (for $n > 1$):

$$\frac{p_1' \quad \cdots \quad p_n'}{\theta p} \quad \begin{cases} \text{where } \neg p_1, \ldots, \neg p_n, p \in \Psi;\ \rho_1, \ldots, \rho_n \text{ are renaming substitutions; and} \\ \theta = \mathrm{mgu}(\langle \rho_1 p_1', \ldots, \rho_n p_n' \rangle, \langle p_1, \ldots, p_n \rangle) \end{cases}$$

The procedure begins with the collection of unit clauses in Ψ and $\neg g$ as the initial set of facts, and succeeds if the empty fact (contradiction) is generated. Because every clause in the theory has a positive literal, the only way an empty fact can be generated is if it proves the fact g itself (note that g is ground). Because this proof starts from the unit clauses and derives newer facts by interpreting the Horn clauses forwards, it is a "bottom-up" variant of the usual Prolog-style logic programming.

Consider the goal sequent in the translation $(\Psi)^o ; [\cdot]_0 \twoheadrightarrow g$ where the atoms are all right-biased. The frontier is every clause $\forall \vec{x}. p_1 \multimap \cdots \multimap p_n \multimap p \in (\Psi)^o$. Focusing on one

[4]Queries with more general goals can be compiled to this form by adding an extra clause to the theory.

such clause gives the following abstract derivation in the forward direction, written using a more transparent notation instead of using the $\mathsf{foc}^{+}_{\Downarrow}$, $\mathsf{foc}^{-}_{\Downarrow}$ and $\mathsf{act}_{\Downarrow}$ relations.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_1 ; [\Delta_1]_{w_1} \twoheadrightarrow p_1}{\Gamma_1 ; [\Delta_1]_{w_1} ; \cdot \twoheadrightarrow p_1 ; \cdot}}{\Gamma_1 ; [\Delta_1]_{w_1} \gg p_1} \quad \cdots \quad \dfrac{\dfrac{\Gamma_n ; [\Delta_n]_{w_n} \twoheadrightarrow p_n}{\Gamma_n ; [\Delta_n]_{w_n} ; \cdot \twoheadrightarrow p_n ; \cdot}}{\Gamma_n ; [\Delta_n]_{w_n} \gg p_n} \quad \dfrac{}{\Gamma ; [\cdot]_0 ; p \ll p}\,\text{rinit}}{\Gamma_1, \ldots, \Gamma_n ; \Delta ; p_1 \multimap \cdots \multimap p_n \multimap p \ll p}\,{\multimap}L}{\Gamma_1, \ldots, \Gamma_n ; [\Delta_1]_{w_1}, \ldots, [\Delta_n]_{w_n} ; all\vec{x}.p_1 \multimap \cdots \multimap p_n \multimap p \ll p}\,\forall L}{\Gamma_1, \ldots, \Gamma_n ; [\Delta_1]_{w_1} \times \cdots \times [\Delta_n]_{w_n} \twoheadrightarrow p}\,\text{delete}$$

In other words, the derived rule is

$$\dfrac{\Gamma_1 ; [\Delta_1]_{w_1} \twoheadrightarrow p_1 \quad \cdots \quad \Gamma_n ; [\Delta_n]_{w_n} \twoheadrightarrow p_n}{\Gamma_1, \ldots, \Gamma_n ; [\Delta_1]_{w_1} \times \cdots \times [\Delta_n]_{w_n} \twoheadrightarrow p}$$

In the case where $n = 0$, i.e., the clause in the Horn theory was a unit clause p, we obtain an initial sequent of the form $\cdot ; [\cdot]_0 \twoheadrightarrow p$. As this clause has an empty left hand side, and none of the derived rules add elements to the left, we can make an immediate observation (lem.6.27) that gives us an isomorphism of rules (thm.6.28).

Lemma 6.27. *Every sequent generated in the proof of the goal* $(\Psi)^{o} ; [\cdot]_0 \twoheadrightarrow g$ *has an empty left hand side.* □

Theorem 6.28 (Isomorphism of rules)**.** *For every clause* $\neg p_1, \ldots, \neg p_n, p \in \Psi$ *there is a derived rule*

$$\dfrac{\Gamma_1 ; [\Delta_1]_{w_1} \twoheadrightarrow p_1 \quad \cdots \quad \Gamma_n ; [\Delta_n]_{w_n} \twoheadrightarrow p_n \quad \theta = \mathrm{mgu}(p_1, \ldots, p_n)}{(\Gamma_1, \ldots, \Gamma_n ; [\Delta_1]_{w_1} \times \cdots \times [\Delta_n]_{w_n} \twoheadrightarrow p)\theta}$$

generated for the proof of the goal sequent $(\Psi)^0 ; [\cdot]_0 \twoheadrightarrow g$ *for a fresh goal literal* g *and only right-biased atoms.*

Proof sketch. Note that only the translations of the Horn clauses are on the frontier. The result follows by a straightforward induction over the structure of a Horn clause and the definition of the $\mathsf{foc}^{+}_{\Downarrow}$, $\mathsf{foc}^{-}_{\Downarrow}$ and $\mathsf{act}_{\Downarrow}$ relations. We omit the details of this rather easy proof that has already been illustrated above. □

These facts let us establish an isomorphism between hyperresolution and right-biased focused derivations.

Theorem 6.29. *Every hyperresolution derivation for the Horn query $\Psi \vdash g$ has an isomorphic focused derivation for the goal sequent $(\Psi)^o \,;\, [\cdot]_0 \longrightarrow\!\!\!\!\rightarrow g$ with right-biased atoms.*

Sketch. For every fact p' generated by the hyperresolution strategy, we have a corresponding fact $\cdot \,;\, [\cdot]_0 \longrightarrow\!\!\!\!\rightarrow p'$ in the focused derivation (up to a renaming of the free variables). When matching these sequents for consideration as input for a derived rule corresponding to the Horn clause $\neg p_1, \ldots, \neg p_n, p$, we calculate the simultaneous mgu of all the p_i and p'_i for a Horn clause, which is precisely the operation also performed in the hyperresolution rule. The required isomorphism then follows from thm. 6.28. □

6.4.4 SLD Resolution

SLD Resolution [61] is a variant of linear resolution that is complete for Horn theories and is the basic reasoning mechanism in Prolog-like logic programming languages. It is sometimes called "top-down" or "goal-directed" logic programming because it starts from the goal literal and reasons backwards to the unit clauses. The procedure is as follows: for the Horn query $\Psi \vdash g$, we start with just the initial clause g, and then perform forward search using the following rule (using Ξ to stand for clauses).

$$\frac{\Xi, q}{(\Xi, p_1[\rho], p_2[\rho], \ldots, p_n[\rho])\theta} \quad \left\{ \begin{array}{l} \text{where } \neg p_1, \ldots, \neg p_n, p \in \Psi\text{; } \rho \text{ is a renaming subst; and} \\ \theta = \text{mgu}(p[\rho], q) \end{array} \right.$$

When $n = 0$, i.e., for unit clauses in the Horn theory, this rule corresponds to simply deleting the member of the input clause that was unifiable with the unit clause (and applying the resulting substitution to the rest of the clauses). The search procedure succeeds when it is able to derive the empty clause.

To show how SLD resolution is modeled by our focusing system, we reuse the translation from before, but this time all atoms are given a left bias. The derivation that corresponds to focusing on the translation of the Horn clause $\neg p_1, \ldots, \neg p_n, p$ is:

$$\dfrac{\dfrac{\dfrac{}{\cdot \,;\, p_1 \gg p_1}\,\text{linit} \quad \cdots \quad \dfrac{}{\cdot \,;\, p_n \gg p_n}\,\text{linit} \quad \dfrac{\dfrac{\Gamma \,;\, [\Delta]_w, p \longrightarrow\!\!\!\!\rightarrow Q}{\Gamma \,;\, [\Delta]_w \,;\, p \longrightarrow\!\!\!\!\rightarrow \cdot \,;\, Q}}{\Gamma \,;\, [\Delta]_w \,;\, p \ll \cdot \,;\, Q}}{\Gamma \,;\, [\Delta]_w, p_1, \ldots, p_n \,;\, p_1 \multimap \cdots p_n \multimap p \ll \cdot \,;\, Q}\,{\multimap}L}{\Gamma \,;\, [\Delta]_w, p_1, \ldots, p_n \longrightarrow\!\!\!\!\rightarrow Q}\,\text{delete}$$

In other words, the derived rule is:

$$\frac{\Gamma\,;[\Delta, p]_w \longrightarrow\!\!\!\!\rightarrow Q}{\Gamma\,;[\Delta, p_1, \ldots, p_n]_w \longrightarrow\!\!\!\!\rightarrow Q}$$

The frontier of the goal $(\Psi)^0\,;[\cdot]_0 \longrightarrow\!\!\!\!\rightarrow g$ in the left-biased setting contains every member of $(\Psi)^0$, so we obtain one such derived rule for each clause in the Horn theory. The frontier contains, in addition, the positive atom g; assuming there is a negative instance of g somewhere in the theory, we will thus generate a single initial sequent, $\cdot\,;[g]_0 \longrightarrow\!\!\!\!\rightarrow g$. We immediately observe that:

Lemma 6.30. *Every sequent generated in the focused derivation of* $(\Psi)^0\,;[\cdot]_0 \longrightarrow\!\!\!\!\rightarrow g$ *is of the form* $\cdot\,;[\Delta]_0 \longrightarrow\!\!\!\!\rightarrow g$. □

Theorem 6.31 (Isomorphism of rules)**.** *For every clause* $\neg p_1, \ldots, \neg p_n, p \in \Psi$*, there is a derived rule*

$$\frac{\Gamma\,;[\Delta, p]_w \longrightarrow\!\!\!\!\rightarrow Q}{\Gamma\,;[\Delta, p_1, \ldots, p_n]_w \longrightarrow\!\!\!\!\rightarrow Q}$$

created for the goal sequent $(\Psi)^0\,;[\cdot]_0 \longrightarrow\!\!\!\!\rightarrow g$ *for some goal literal* g *and only left-biased atoms.*

Proof sketch. Note that only the translations of the clauses and the goal literal g itself are in the frontier. For g, we get just a single initial sequent $\cdot\,;[g]_0 \longrightarrow\!\!\!\!\rightarrow g$. For the translation of the clauses, we use a simple induction on the structure of the clauses and the definition of the $\mathsf{foc}^+_\Downarrow$, $\mathsf{foc}^-_\Downarrow$ and $\mathsf{act}_\Downarrow$ relations. Again, we omit the rather easy proof that has been illustrated above. □

Theorem 6.32. *Every SLD resolution derivation for the Horn query* $\Psi \vdash g$ *has an isomorphic focused derivation for the goal sequent* $(\Psi)^o\,;[\cdot]_0 \longrightarrow\!\!\!\!\rightarrow g$ *with left-biased atoms.*

Proof sketch. Very similar argument as in thm. 6.29, except we note that in the matching conditions in the derived rules we rename the input sequents, whereas in the SLD resolution case we rename the Horn clause itself. However, this renaming is merely an artifact of the procedure and doesn't itself alter the derivation. □

Although the derivations are isomorphic, the focused derivations may not be as efficient as the SLD resolution in practice because of the need to rename (i.e., copy) the premisses as part of the matching conditions of a derived rule– premisses might contain many more components than the Horn clause itself.

6.4.5 Historical review

The concept of viewing focused derivations as a means of constructing derived inference rules is not new. Andreoli himself has made similar attempts for backward reasoning: see [8], for instance. Girard's *Ludics* [44] uses focusing as a foundational concept and takes it as an explanation for logic; in Ludics, "bipoles" or derived inference rules are the only rules that are syntactically allowed. Focusing for intuitionistic (including linear) logics was first investigated by Howe [57]; however, Howe was not aware of the notion of focusing bias, and his calculus furthermore had certain technical oddities (such as possibly infinite loops) that have been corrected in this work with a careful treatment of atomic propositions. The precise combination of focusing with forward reasoning in the inverse method is also a contribution of this thesis; prior to this work, such combinations existed in the domain of conjecture and folklore. There are significant details in the construction of the focusing calculus and its use in generating forward derived inference rules.

The quest for ways of making large, multi-step derivations has a long history in automated reasoning. As mentioned earlier, hyperresolution itself is one such logically motivated strategy. Other strategies such as chaining single-premiss rules or using monadic properties of the logic have also been attempted [112]. These heuristics are not as far reaching as focused derivations, which, as demonstrated in the previous section, subsumes hyperresolution on Horn theories.

The interaction of focusing and cut-elimination has been studied by Danos, Joinet and Schellinx [33, 32]. Although none of their translations are explicitly focusing aware, their calculi, particularly the constraints in the LK_p^η system bear unmistakable similarities to focusing. A more recent work by Jagadesan *et al* [58] is the system λRCC, a logic programming language without focusing, but with the notion of biased atoms. In λRCC the observation that switching the bias gives rise to forward- or backward-chaining is certainly visible, though this observation is limited to the Horn-fragment of intuitionistic logic.

Ideas of polarity and focused derivations are increasingly becoming a standard technique in type theory. Laurent [65, 64] has used the notion of polarity to explain type isomorphisms in call-by-value and call-by-name settings and gives algorithms for generating dual program constructs in terms of a Curry-Howard isomorphism. At least the asynchronous half of focusing has been (probably independently) noted by Levy [66] in the domain of operational semantics for programming languages, though his calculus is not as powerful as focusing.

Future work

The main open question raised by the previous section is whether the observation that focusing generalises hyperresolution and SLD resolution on the Horn fragment can be extended to a fuller logic. This question is naturally meaningless for intuitionistic logic because hyperresolution is a classical strategy. Focusing for purely classical proof search is also not very satisfactory (though see a recent attempt to do just that [119]). For classical linear logic, however, the question is an interesting one. We conjecture that the focusing calculi already available for classical linear logic, or if necessary an adaptation of the intuitionistic focusing calculus of this work, will turn out to give an explanation for classical hyperresolution.

Another important item of future work would be a detailed analysis of connections with a bottom-up logic programming interpreter for the LO fragment of classical linear logic [18]. This fragment, which is in fact affine, has the property that the unrestricted context remains constant throughout a derivation, and incorporates focusing at least partially via a back-chaining rule. It seems plausible that our prover might simulate their interpreter when LO specifications are appropriately translated into intuitionistic linear logic, similar to the translation of classical Horn clauses.

Chapter summary *We have reconstructed focused derivations for intuitionistic linear logic and extended it with a notion of focusing bias for atomic propositions.. We have given a novel completeness proof for this calculus in terms of cut-admissibility and the identity principle. We have then presented a design of an inverse method theorem prover based on forward derived inference rules constructed from focusing.*

Additionally, we have shown how forward focusing can be understood as a generalisation of both hyperresolution and SLD resolution for Horn theories, depending on a choice of focusing bias. Furthermore, we have shown a new translation of intuitionistic logic to linear logic that is faithful to a focused sequent calculus for the source logic.

Chapter 7

Experimental evaluation

In this chapter we present the LI (standing for "Linear Inverse") family of theorem provers that implement the inverse method for various fragments of linear logic presented in this thesis. The provers will either using single inference rules like in chapters 4 or 5, or derived inference rules using focusing as in chapter 6. All versions of the prover use a uniform input syntax and output format, though a prover for a more restricted fragment will not be able to handle elements in the input that it has no way of handling. The fragments of the logic and the strategies used by the provers will be indicated using affixes to its name.

This chapter is organized as follows: in section 7.1 we present the syntax for the LIF prover, which has the most complete input language. In sec. 7.2 we shall present experiments with the purely propositional prover LIP. In sec. 7.3 we shall look at general first-order linear problems. In sec. 7.4 we shall consider theorem proving problems in other logics in translation to linear logic.

Experimental framework All experiments on the LI provers were done using MLTon version 20060213. For the propositional experiments, some results of running the same experiments on the Gandalf "nonclassical" distribution (version 0.2), compiled using its own packaged version of the Hobbit Scheme compiler, are presented. This distribution contains two propositional linear logic provers: one using resolution (denoted as **Gr** in this text), and the other using a Tableaux representation (written `Gt`). We did not attempt to bound the search for either version of the Gandalf prover; neither did we alter any of the default runtime parameters. Other provers such as LinTAP [70] and llprover [113] fail to prove all but the simplest problems, so we did not do any serious comparisons against them. Our experiments were all run on a 3.4GHz Pentium 4 machine with 1MB L1 cache and 1GB main memory. Most running times are wall clock times, but they are averaged over five runs for extremely short running times (less than 0.01 seconds).

7.1 External syntax

Every LI prover accepts a textual file with a list of directives and declarations as input, and executes the directives in the order that it finds them. In addition, the external syntax gives a textual representation of propositions and proof terms. Terms are either variables or function symbols applied to a list of argument terms, written using the standard tuple notation, i.e., of the form `f(t`$_1$`, ..., t`$_n$`)`. If the list of arguments is empty, then it is simply left out and the function symbol represents a constant. Lexically, both variables and function symbols are any alphanumeric identifier starting with a letter; no distinction is made between variables and function symbols.

For propositions, we use the lexemes `*`, `1`, `-o`, `&`, `#`, `+`, `0` and `!` to stand for $\otimes$, $\mathbf{1}$, $\multimap$, $\&$, $\top$, $\oplus$, $\mathbf{0}$ and ! respectively. For the quantifiers, `(x)A` represents $\forall x.A$ where `x` and `A` are representations of x and A respectively; similarly `[x]A` for $\exists x.\, A$. Atomic propositions are treated in the same way as terms, and there is no lexical distinction made between a predicate symbol and a function symbol. In the extensions of LI with the modal operators, we add two additional syntactic forms: `?A` for the possibility modal proposition $?\,A$, and `{A}` for the lax modal proposition $\{A\}$. Finally, we add the following derived forms:

1. `A -> B` and `B <- A` for $!A \multimap B$
2. `A == B` for $(A \multimap B) \mathbin{\&} (B \multimap A)$

The binary connectives are written in an infix style with the following order of precedence (from lowest to highest): `==`, (`<-`, `o-`), (`->`, `-o`), `+`, `&`, `*`. The connectives `-o` and `->` associate to the right, and all other binary connectives associate to the left. Quantifiers have the lowest precedence, and unary operators have the highest precedence.

In addition to propositions, there is also external syntax for describing proof terms. The syntax of proof terms allows writing normal proof terms generally, but it also allows for explicit coercions for non-normal proof terms. The grammar for these proof terms is

```
(inferable terms)     i  ::= u                                     (hyp)
                         |  p                             (defined term)
                         |  i c                                    (⊸E)
                         |  fst i | snd i                          (&E)
                         |  (c:A)                         (coercion c:A)

(checkable terms)     c  ::= i                                  (silent)
                         |  c * c                                  (⊗I)
                         |  let u * v = i in c end                 (⊗E)
                         |  1                                      (1I)
                         |  let 1 = i in c end                     (1E)
                         |  \u. c                                  (⊸I)
                         |  (c, c)                                 (&I)
                         |  ()                                     (⊤I)
                         |  inl c | inr c                          (⊕I)
                         |  case i of inl u => c | inr v => c'     (⊕E)
                         |  abort i                                (0E)
                         |  ! c                                    (!I)
                         |  let !u = i in c end                    (!E)

                         |  let u = i in c end    (linear substitution [i/u]c)
                         |  ulet u = i in c end  (unrestr. substitution [i/u]c)
```

Figure 7.1: BNF for proof terms

essentially the textual representation of the definition in sec. 2.2.3, i.e., the BNF shown in figure 7.1. The syntax also admits two let-forms to represent the substitution principles in the logic. In this figure the modal operators (and their corresponding let-forms) are not shown, but they will be described later.

The input file consists of a sequence of declaration and prover directives; the former defines new symbols in the prover, and the latter triggers various searches, computations, and checks in the prover. In the rest of this section we shall summarise the key features.

Propositional declarations New propositions can be defined using the syntactic form `p : A`. Such declarations are automatically treated as global hypotheses for the prover. The left hand side of the colon defines a new label for the hypothetical proof of this proposition, and the right defines the proposition that is being assumed. A common use of such propositional declarations is to specify the problem domain or theory in which queries are to be made. These propositional declarations are automatically globalised (see sec. 6.3.1).

Proof search To search for the proof of a proposition, we use the `%prove` directive:

```
%prove A.
```

This triggers a search for the proof of `A` with every proposition declared earlier present in the unrestricted context of the goal sequent. This simple declaration can be further refined by specifying bounds on the number of iterations; the following, for example, attempts 2000 iterations on `A` and then saturates.

```
%prove 2000, A.
```

If the proposition is false, then we can use the `%refute` or `%saturate` declaration to attempt to saturate the search space. Refutation can also be bounded in the number of iterations, but in this case after the saturation *succeeds* early.

If a proof is found, then the corresponding proof term is printed. This proof term can also be captured in the form of a proof term definition using the variant form

```
%prove p :  A.
```

The found proof is bound to `p`, which can then be used in future operations on proof terms.

Proof normalisation Given a non-normal proof term c, i.e., a proof term that uses coercions, its normal form can be computed using the `%norm` declaration.

```
%norm d = c.
```

The normal form of `c` is computed and bound to the symbol `d`.

Proof checking Given a normal proof term, it can be checked against a given proposition using the `%check` directive.

```
    %check c :  A.
```

If the proof `c` checks against the proposition `A`, then check is successful and LI prints "OK"; otherwise it terminates with an error message. The dual of `%check` is `%reject`, with the same arguments, that verifies that the given proof term is *not* a proof of the given proposition.

Note that any proof term may use the terms defined by `%prove` and `%norm`, and the names of the declared propositions. However, the semantics of such uses is a bit odd: internally, the definitions of the constants are expanded and turned into a coercion, then normalised. Thus, the resulting proof term may be very different from what the user writes down depending on the details of the normalisation algorithm. Nevertheless, we prefer this approach because of the well-behaved bidirectional proof-checking algorithm that allows us to omit most of the type (proposition) annotations out of proof terms; indeed, the η-long and β-normal terms need no type information at all.

In addition to these logical declarations and definitions, there is an elaborate logging facility present in LI for recording various statistics and intermediate stages of proof search. The details of these facilities are not very relevant to this thesis, but can be found in the documentation accompanying the LI distribution.

7.2 Propositional experiments

The first batch of experiments use a purely propositional prover using the calculus of chapter 4, named LIP, and its focusing variant, named LIPF. Both versions of this prover use the extra optimisations available in the propositional case such as irredundant rules (see sec 3.4) and efficient propositional contraction.

7.2.1 Planning

Blocks world is a simple planning example that can be embedded entirely into the multiplicative-exponential fragment (i.e., {$\otimes$, $\multimap$, !}) of linear logic. For our experiments we consider a simple world with exactly three blocks named a, b and c. The robotic arm is represented by four atomic propositions: `empty` for the empty arm, and `holds_a`, `holds_b` and `holds_c` for holding a, b and c respectively. The state of blocks not held by the robotic arm depicts the placement of blocks. For example, `on_a_b` represents the state that the block a is on b, and `on_a_tab` depicts that a is on the table. The arm is only allowed to lift blocks that have no blocks on top of them; this condition is represented by the three atoms `free_a`, `free_b` and `free_c`.

Actions by the robot depict a transformation of the state. For instance, if the arm lifts the block a from the block b, then that state change is:

```
pick_a_form_b: free_a * on_a_b * empty -o free_b * holds_a.
```

The reverse is also a valid action:

```
drop_a_on_b: holds_a * free_b -o on_a_b * free_a * empty.
```

There are thus a total of 18 such rules.

We start from an initial state where a is on b, and b and c are on the table and the hand is empty (fig. 7.2 (a)). This is represented by:

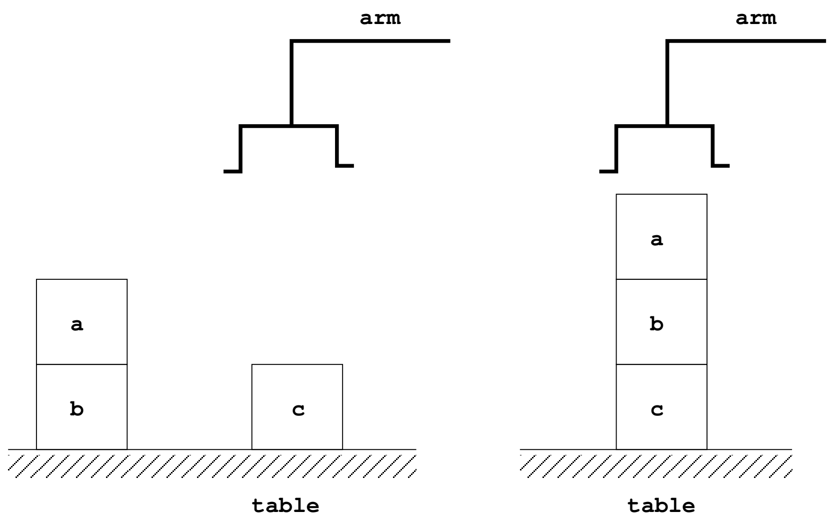
```
initial = on_a_b * free_a * on_b_tab * on_c_tab * free_c * empty.
```

From this state we want to move the (a, b) tower on top of c (fig. 7.2 (b)):

```
final = on_a_b * on_b_c * on_c_tab * #.
```

$\top$ (#) is used because we don't really care about the rest of the final state. To instruct the prover to prove this, we issue the following directive.

```
%prove initial -o final.
```



(a) start state (b) end state

Figure 7.2: A blocks world problem

This variant we call `blocks`, the purely linear variant. We also attempt the same exercise using the lax modality {} using a variant of these rules where the result is in the monad. For example:

```
pick_a_from_b_monadic: on_a_b * free_a * empty -o {free_b * holds_a}.
```

Here we direct the prover as follows:

```
%prove initial -o {final}.
```

The use of the monad makes the sequence of actions explicit in the proof. This makes the proofs have a focusing flavour even in the absence of a focusing calculus. Once a rule is used with $\multimap L$, the proof is automatically focused on the succeedent of the implication. The focusing calculus of course has a similar nature even in the absence of the CLF monad, though the natures are not precisely identical.

In experiments, we found that this problem was large enough that in the absence of focusing it runs for a few hours before exhausting the memory of the experimental system after about 400,000 iterations of the lazy OTTER loop and about 70,000 generated sequents. In looking at traces of the run, we found that there were simply too many intermediate sequents generated for every use of a state transition rule, and the transitions were being attempted too often. Furthermore, because the rules `pick_a_from_b` and `drop_a_on_b`, for example, are inverses of each other, the prover gets mired in more and more complex cycles of actions without making progress.

For the focusing prover LIPF, the situation is drastically different. Not only are the proofs found quickly, but also the CLF monad appears to give no benefits (and in fact adds slightly to the overhead). Here "iters" refers to the number of iterations of the OTTER loop, "gen" for the number of generated sequents, "subs" for the number of forward-subsumed sequents, and "time" for the wall-clock times in seconds. Note that the Gandalf provers do not have support for the CLF monad, and fail to prove the non-monadic versions.

	right-biased				**left-biased**				`Gt`	**Gr**
problem	iters	gen	subs	time	iters	gen	subs	time	time	time
`blocks`	45	424	317	0.12	26	387	337	**0.04**	×	×
`blocks-clf`	64	697	412	0.264	15	81	69	**0.006**	N/A	N/A

Here we observe that left-biasing is about an order of magnitude faster than right-biasing. The reason for this is that all derived rules in the left-biased system are single premiss rules, so there is no overhead due to the percolation phase (defn. 5.30).

We have also considered other planning problems that can be expressed in this multiplicative-exponential fragment, and they appear to have similar results. The first of these is a change-making problem that defines the transformations in a change machine. This domain has rules like

```
quarter -o dime * dime * nickel.
dime -o nickel * nickel.
nickel -o penny * penny * penny * penny * penny.
```

Queries in this theory ask if a given initial collection of coins can be converted to another given collection, and from the proofs of these queries we can extract the actual steps used to make this transformation. The table below documents the results of a particular problem that is feasible without focusing.

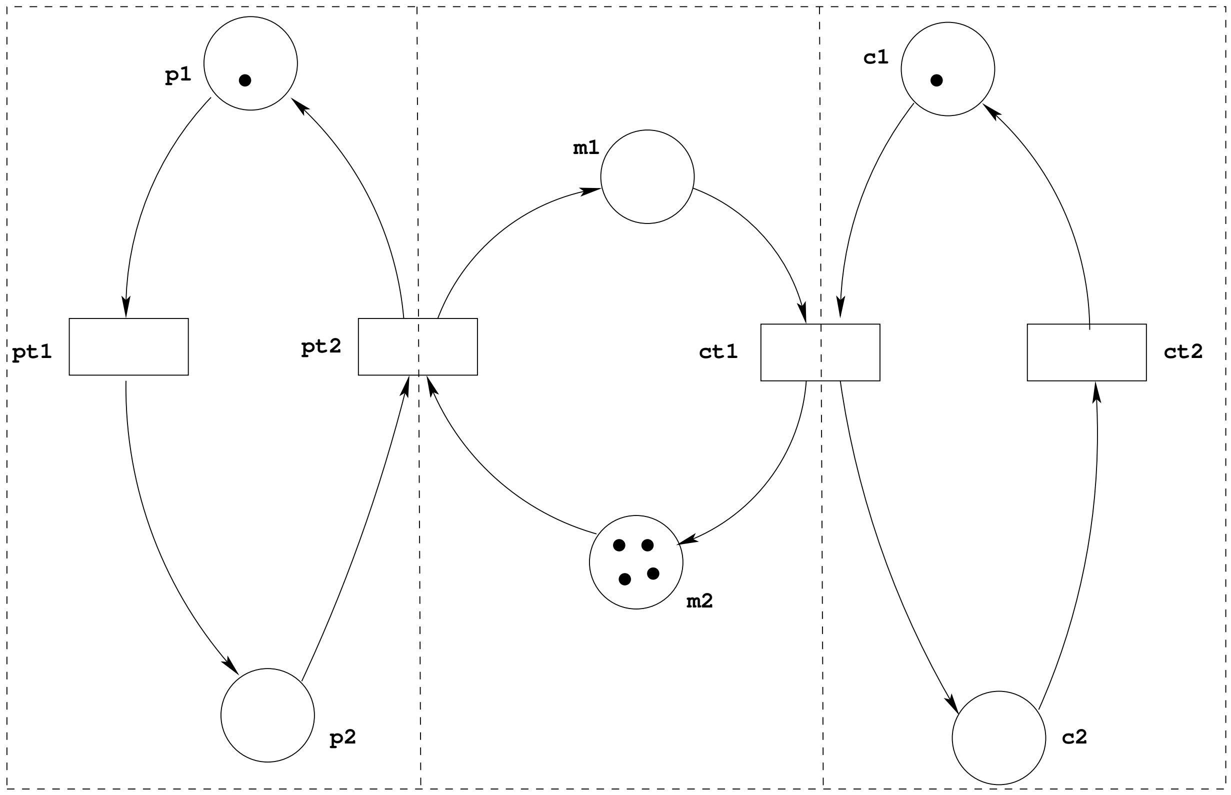


Figure 7.3: A producer-mediator-consumer net

	LIP	right-biased LIPF				left-biased LIPF				Gt	Gr
name	time	iters	gen	subs	time	iters	gen	subs	time	time	time
change	3.196	16	22	7	0.001	11	20	6	0.001	0.63	0.31

For most problems the focusing bias in LIPF seems to have only a minor impact on the running time. The drastic difference comes instead from using any focusing system at all.

The last of these planning problems we considered was encodings of Petri-nets [25]. The table below contains the total running time for several different Petri nets, of which we shall describe just one in some detail. This example net is shown in figure 7.3. It is a *producer-consumer* network: the left half, the producer of tokens, synchronises with the right half, the consumer, using a mediating buffer m1 and m2. The number of tokens in this buffer represent the maximum number of cycles that the producer and consumer can differ by.

Each place and transition in the network is represented in terms of an atomic propositions. The two producer places are `p1` and `p2`, and their transitions are `pt1` and `pt2`. The first transition `pt1` is enabled whenever there is a token in `p1`, so we write this as

```
p1 -o pt1.
```

This transition produces a token in `p2`, which we write as:

```
pt1 -o p2.
```

The second transition `pt2` is enabled whenever there is a token in `p2` and `m2`, and it produces a token each in `p1` and `m1`. We write these rules as follows:

```
p2 * m2 -o pt2.
pt2 -o p1 * m1.
```

The consumer is similarly described:

```
c1 * m1 -o ct1.          ct1 -o m2 * c2.
c2 -o ct2.               ct2 -o c1.
```

The particular initial state with a mediating buffer of size 4 shown in figure 7.3 is:

```
initial = p1 * m2 * m2 * m2 * m2 * c1.
```

It represents the state where the consumer has just consumed all four tokens that the producer has previously produced. We ask the query if in a future state the producer can have produced four more tokens.

```
final = m1 * m1 * m1 * m1 * #.
%prove initial -o final.
```

Like before with the blocks world, we also include a monadic version of the encoding where the succeedent of the state transformations are always inside the monad and the final query is `initial -o {final}`. The table below summarises the results. Note that the size of these problems is large enough that the small step prover LIP exhausts the system memory easily; thus we present only the data for the focusing prover LIPF. The examples `petri-1` all ask for provable queries, whereas `petri-2` are all false queries for which we ask for (unbounded) saturation. (The same Petri nets are used in both.)

	right-biased LIPF				**left-biased LIPF**				**Gt**	**Gr**
name	iters	gen	subs	time	iters	gen	subs	time	time	time
`petri-1`	23	38	23	**0.001**	284	1099	921	0.062	×	7.08
`petri-2`	57	133	105	**0.003**	393	1654	1433	0.068	×	7.13

Here we observe that right-biasing performs better than left-biasing. One indication that this would be the case is in the structure of the rules which all have either a singleton antecedent or a singleton succeedent; the state changes are therefore more structured than arbitrary multiset rewriting for which left-biasing would be a better approach.

7.2.2 Graph exploration

The next class of propositional examples are graph exploration algorithms encoded in linear logic with the additive connectives. Our graphs are always directed, but can be cyclic. The first of these algorithms attempts to find an Euler tour in the graph if one exists. To implement this algorithm, we represent the edges of the graph as linear implications between the vertices, but these implications are themselves only allowed to be used linearly. Thus, given a starting vertex, if all the edges in the graph can be consumed to end back that same vertex, then there exists an Euler tour in the graph (which can be extracted from the proof).

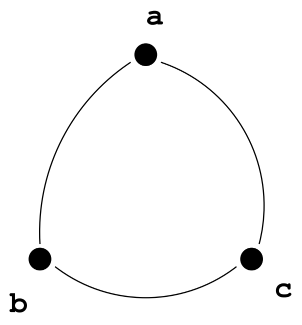


Figure 7.4: A simple graph

To give a simple example, consider the triangular graph in figure 7.4. It has three vertices a, b and c, and the edges between the graph are:

```
edge_a_b = (a -o b) & (b -o a).
```

```
edge_a_c = (a -o c) & (c -o a).
edge_b_c = (b -o c) & (c -o b).
```

To ask if there is an Euler tour starting at a, we issue the following directive:

```
%prove edge_a_b * edge_a_c * edge_b_c -o a -o a.
```

In the following table, we summarise the results of asking for Euler tours in the complete graphs of up to 10 vertices. The set euler-1 goes up to size 6, the next set euler-2 between 7 and 9 vertices, and euler-3 for the complete graph of 10 vertices. Needless to say, the small step prover LIP fails to prove any of them, so we once again present only the focusing results.

	right-biased LIPF				**left-biased LIPF**			
name	iters	gen	subs	time	iters	gen	subs	time
euler-1	6291	11853	5565	9.010	6291	11853	5565	**8.570**
euler-2	15640	34329	18689	152.12	15640	34329	18689	**145.9**
euler-3	64360	159194	94834	3043.35	64360	159194	94834	**2938.55**

Because the encoding of the theory is completely symmetric, both backward and forward chaining generate the same sequents. Interestingly, a left-biased search performs slightly better than the right-biased system because of peculiarities of the implementation that produces the sequent in a slightly different order in each case.

The other graph exploration algorithm we consider is finding Hamiltonian tours; its design is very similar. Here, for each vertex we maintain two states: visited, or unvisited (indicated using primes). If the vertex a is connected to b, then the transition between them is allowed only if b has not been visited before, and as the result of the transition b becomes visited. We write this as

```
visit_a_b = a * b' -o b.
```

Of course, this particular edge might not be used in a Hamiltonian tour, so we optionally allow it not to be used:

```
visit_a_b = (a * b' -o b) & 1.
```

Consider the graph in figure 7.4 again, but now direct the edges from `a` to `b`, from `b` to `c` and from `c` to `a`. In addition to the above rule, we have the following rules for the remaining vertices.

```
visit_c_a = (c * a' -o a) & 1.
visit_b_c = (b * c' -o c) & 1.
```

We start out with all vertices unvisited

```
initial = a' * b' * c'.
```

And from this we ask if a complete tour exists starting and returning to a given vertex, `a`:

```
%prove visit_a_b * visit_b_c * visit_c_a -o initial -o (a -o a).
```

In the following table we summarise the results of asking these queries on all compete graphs of up to 8 vertices. We also ask the prover to saturate on a few directed acyclic graphs of size up to 4.

	right-biased LIPF				**left-biased LIPF**			
name	iters	gen	subs	time	iters	gen	subs	time
`hamilton`	708	911	185	0.11	165	178	0	<**0.001**

The surprising result in these examples is that left-biasing is vastly superior to right-biasing. Examination of the derived rules reveals that all rules in the left-biased case for the edge transitions are single-premiss rules, whereas they have two premisses each for the right-biased system. Furthermore, in the right biased case there is an overlap in the rules as some of the premisses are satisfied by implicit weakening from the affine context, as presented in section 3.3. All the sequents generated in this fashion were immediately subsumed, but the redundancy had a noticeable overhead. The left-biased system, on the other hand, proceeds methodically to explore the graph, making not a single redundant choice. The sequents generated in this case contain a pleasing visualisation of the current "state" of the exploration of the graph.

7.2.3 Affine logic problems

Linearity is often too stringent a requirement for situations where we simply need *affine* logic, i.e., where every hypothesis is consumed *at most* once. Affine logic can be embedded into linear logic by translating every affine arrow $A \to B$ as either $A \multimap B \otimes \top$ or $A \mathbin{\&} \mathbf{1} \multimap B$. Of course, one might select complex encodings; for example choosing $A \mathbin{\&} !(\mathbf{0} \multimap X) \multimap B$ (for some arbitrary fresh proposition X) instead of $A \mathbin{\&} \mathbf{1} \multimap B$. Even though the two translations are equivalent, the prover performs poorly on the former. The Gandalf provers **Gt** and **Gr** fail on these examples (give incorrect answers).

	right-biased				**left-biased**			
encoding	iters	gen	subs	time	iters	gen	subs	time
$A \multimap B \otimes \top$	38	108	73	0.003	34	107	73	**0.002**
$A \mathbin{\&} \mathbf{1} \multimap B$	252	1103	828	0.098	62	229	126	**0.019**
$A \mathbin{\&} !(\mathbf{0} \multimap X) \multimap B$	264	7099	6793	2.028	235	841	578	**0.042**

It should be noted that, as mentioned in sec. 3.3, that our treatment of negative $\&\mathbf{1}$ is merely a heuristic, as the question of when a proposition is equivalent to $A \mathbin{\&} \mathbf{1}$ is itself as hard as proving arbitrary theorems (and therefore undecidable). However, if the user were to carefully use $\&\mathbf{1}$ instead of $\& !(\mathbf{0} \multimap X)$, then this heuristic suffices.

7.3 First-order experiments

7.3.1 First-order planning

In this section we shall have the first-order version of the blocks world problems in sec. 7.2.1. With quantifiers, it becomes considerably easier to specify the rules of the system, as one can write them generically about all blocks. In this encoding, the blocks and the table become terms. The state of the robotic arm is represented in terms of two atoms: the propositional atom `empty` from before, and the predicate `holds()` that takes the block it holds as an argument. Similarly, `on()` becomes a binary predicate, and `free()` a unary predicate.

The "pick" rule is written generically by quantifying over all blocks:

```
    pick_from_block: (a) (b) on(a,b) * free(a) * empty
                                  -o holds(a) * free(b).
```

The table can be implemented in two ways. The first way treats the table as another block that is always free: this it accomplishes by asserting `free(table)` as a new unrestricted resource, and requires a new rule, `free(table) -o 1`, for "consuming" `free(table)` that gets created by a rule such as `pick_from_block`. We don't take this approach because negative propositions of the form $A \multimap 1$ in the forward direction ends up blowing up the sequent database because of reasons outlined in sec. 3.3. Instead, we add a new rule for picking up blocks from the table.

```
    pick_from_table: (a) on_table(a) * free(a) * empty -o holds(a).
```

The particular example from sec. 7.2.1 becomes the following directive in the first order case.

```
    %prove  on(a,b) * on_table(b) * on_table(c)     %
            * free(a) * free(c) * empty             % Initial state
            -o on(a,b) * on(b,c) * #.               % Final state
```

We also include the monadic version of this problem with the succeedents of implications in the {} monad.

The following table summarises the results of our experiments on one such query. Both LIF and LIFF implement globalisation (sec. 6.3.1), without which the non-focusing prover LIF takes almost ten times as long to complete.

	LIF	right-biased LIFF				left-biased LIFF			
problem	time	iters	gen	subs	time	iters	gen	subs	time
`blocks`	0.036	45	424	317	0.12	26	387	337	**0.04**
`blocks-clf`	0.046	64	697	412	0.264	15	81	69	**0.006**

As remarked already in sec. 7.2.1, the left-biased system strongly outperforms the right-biased system. Six actions are required by the robotic arm to achieve the desired result; on observing the sequents generated during the search, it appears that the left-biased LIFF comes requires exactly two iterations of the OTTER loop per action in the monadic case.

Another simple planning problem we considered in the first order case is Dijkstra's urn game. In this game, there is an urn containing black and white balls. In each round two balls are removed from the urn. If both balls are the same colour, then a black ball is added back to the urn; otherwise, the black ball is discarded and the white ball added back to the urn. This process is repeated until only a single ball remains in the urn.

The encoding of this game is extremely easy: represent the balls with the predicate `ball()`, and their colours by the terms `black` and `white`. The multiplicity of `ball(black)` denotes the number of black balls in the urn; similarly for `ball(white)`. The transition rules are:

```
same: (c) ball(c) * ball(c) -o ball(black).
diff: ball(white) * ball(black) -o ball(white).
```

An example problem is:

```
%prove ball(black) * ball(white) * ball(white) * ball(black)
        -o ball(white).
```

The following table summarises the results of several such example problems.

	LIF	right-biased LIFF				left-biased LIFF			
problem	time	iters	gen	subs	time	iters	gen	subs	time
urn	3.08	29	72	27	0.24	13	58	55	**0.11**

The problems included both satisfiable and unsatisfiable problems. Unfortunately, in the unsatisfiable case the prover loops forever instead of saturating, as there is no external imposition on the maximum number of balls in the urn. That is to say, although the rules guarantee that the number of balls decreases in each step, this fact is not exploitable in the inverse method where every sequent reasons only about a fraction of the state.

7.4 Translations to linear logic

In this section we examine the performance of the linear inverse method as a reasoning framework for other logics that can be embedded into linear logic. There are two main motivations for this examination. First, that it gives us a ready source of difficult problems to stress test the prover. The translations from QBFs to linear logic in sec. 7.4.1, for example, were used to show that propositional multiplicative additive linear logic is PSPACE-complete [68].

The second benefit to looking at translations is that it gives us an indication of the utility of linear reasoning in a setting where many of the problems will be non-linear. Ideally, the price of linear reasoning should not be so high that it precludes any non-linear reasoning. In particular, any focusing features of the non-linear logic should be expressible in terms of the focusing present in the linear logic.

7.4.1 Quantified Boolean formulas

As shown by Lincoln *et al* in [68], propositional multiplicative additive linear logic is sufficiently powerful to embed quantified Boolean formulas[39]. The key idea of the embedding is to interpret the quantified Boolean variables as signals in a circuit, and use the linear connectives to give an interpretation of the Boolean connectives in terms of signals. The algorithm in [68] was given for classical linear logic, but it is easily adapted to the intuitionistic case, and in fact gives more perspicuous interpretations. In this section we will look at two minor variants of this embedding: one that is in the pure multiplicative-additive fragment, and one that uses a small number of exponentiated implications in order to propagate the symbols. While the logic used in the latter embedding will be far more expressive (and undecidable) than that of the former, the encoding will be simpler and the theorem prover LIPF will perform better on it.

The language of the QBFs will have the binary connectives $\vee$, $\wedge$, the unary connective $\neg$, the propositional constants $\top$ and $\bot$, and the propositional quantifiers $\forall$ and $\exists$. QBFs will be written using lowercase letters $p, q, \ldots$ and variables with the x, y, etc. We assume that all formulas are *rectified*, i.e., all bound variables in the formula are distinct.

Definition 7.1 (Multiplicative additive embedding).
The output *of a rectified quantified Boolean formula along a constant, written $\langle - \rangle_{-}$, is defined inductively over the structure of of the proposition and obeys the following equations.*

$$\langle x \rangle_s = (x \multimap x \otimes s) \mathbin{\&} (\overline{x} \multimap \overline{x} \otimes \overline{s})$$

$$\langle \top \rangle_s = s \qquad \langle \bot \rangle_s = \bar{s}$$

$$\langle \neg p \rangle_s = \langle p \rangle_{s'} \otimes \Big((\overline{s'} \multimap s) \mathbin{\&} (s' \multimap \bar{s}) \Big) \qquad s' \text{ and } \overline{s'} \text{ fresh}$$

$$\left.\begin{aligned} \langle p \wedge q \rangle_s &= \langle p \rangle_{s_1} \otimes \langle q \rangle_{s_2} \otimes \texttt{conj}(s_1, s_2, s) \\ \langle p \vee q \rangle_s &= \langle p \rangle_{s_1} \otimes \langle q \rangle_{s_2} \otimes \texttt{disj}(s_1, s_2, s) \end{aligned}\right\} \qquad s_1, \overline{s_1}, s_2, \text{ and } \overline{s_2} \text{ fresh}$$

$$\langle \forall x.p \rangle_s = \Big(x \otimes (x \multimap \mathbf{1}) \otimes \langle p \rangle_s \Big) \oplus \Big(\bar{x} \otimes (\bar{x} \multimap \mathbf{1}) \otimes \langle p \rangle_s \Big)$$

$$\langle \exists x.p \rangle_s = \Big(x \otimes (x \multimap \mathbf{1}) \otimes \langle p \rangle_s \Big) \mathbin{\&} \Big(\bar{x} \otimes (\bar{x} \multimap \mathbf{1}) \otimes \langle p \rangle_s \Big)$$

Here "fresh" means that the indicated signal occurs nowhere outside the body of the translation. The pair `conj` *and* `disj` *encode the truth tables for* $\wedge$ *and* $\vee$ *respectively. Precisely,*

$$\begin{aligned} \texttt{conj}(s_1, s_2, s) = & \quad (s_1 \multimap s_2 \multimap s) \\ \& & \quad (s_1 \multimap \overline{s_2} \multimap \bar{s}) \\ \& & \quad (\overline{s_1} \multimap s_2 \multimap \bar{s}) \\ \& & \quad (\overline{s_1} \multimap \overline{s_2} \multimap \bar{s}) \\ \texttt{disj}(s_1, s_2, s) = & \quad (s_1 \multimap s_2 \multimap s) \\ \& & \quad (s_1 \multimap \overline{s_2} \multimap s) \\ \& & \quad (\overline{s_1} \multimap s_2 \multimap s) \\ \& & \quad (\overline{s_1} \multimap \overline{s_2} \multimap \bar{s}) \end{aligned}$$

The translation of the quantified Boolean formula p, written $\langle p \rangle$, *is then* $\langle p \rangle_s \multimap s$ *where s and* $\bar{s}$ *are fresh.*

To get a rough idea for why this translation works, note that the proposition $x \otimes (x \multimap \mathbf{1})$ denotes the assumption that x is true, and $\bar{x} \otimes (\bar{x} \multimap \mathbf{1})$ that x is false. For a universal quantification $\forall x.p$, every use of x in p will turn into $(x \multimap x \otimes s) \mathbin{\&} (\bar{x} \multimap \bar{x} \otimes \bar{s})$. If x is true, then the left operand of $x \otimes (x \multimap \mathbf{1})$ will match up with the left operand of $(x \multimap x \otimes s) \mathbin{\&} (\bar{x} \multimap \bar{x} \otimes \bar{s})$ to produce $x \otimes s$, of which the left half x will again be removed by $x \multimap \mathbf{1}$, leaving just s at the sites where x occurred in p. Similarly, if x is false, then at these occurrences we will be left with $\bar{s}$. This sketch can be formalised to give an embedding theorem whose proof we leave as an exercise.

Fact 7.2 (Embedding)**.**
For a closed quantified Boolean formula p,

1. *p is true if and only if $\cdot\,;\langle p\rangle_s \Longrightarrow s$; and*
2. *p is false if and only if $\cdot\,;\langle p\rangle_s \Longrightarrow \overline{s}$.*

In our implementation, we translated several QBFs using this translation and attempted to prove the resulting sequent. The QBFs we looked at had between 2 and 5 variables and between 1 and 3 quantifier alternations. The largest example in the test suite was the translation of the transitivity of implication, $\forall x.\forall y.\forall z.(x \supset y) \supset (y \supset z) \supset (x \supset z)$, where $p \supset q$ was defined as $\neg\, p \vee q$ (`qbf-3`). The results are summarised in the following table.[1]

	right-biased LIPF				**left-biased LIPF**			
encodings	iters	gen	subs	time	iters	gen	subs	time
`qbf-1`	1457	5590	4067	**0.54**	1581	4352	2612	0.58
`qbf-2`	15267	517551	502174	368.92	9469	49777	37716	**29.55**
`qbf-3`	28556	990196	961494	2807.64	21233	89542	115917	**308.24**

As the size of the example increases, the left-biased system overtakes the right-biased system and eventually becomes nearly an order of magnitude faster.

The second QBF translation we used is a slight variant of the above translation, except we allow for arbitrary copying of signals instead of depending on $x \multimap 1$ etc. to remove the excesses.

Definition 7.3 (Exponential embedding).
The output *of a rectified quantified Boolean formula along a constant, written $\langle\!\langle - \rangle\!\rangle_{-}$, is defined inductively over the structure of of the proposition and obeys the following equations.*

$$\langle\!\langle x \rangle\!\rangle_s = (x \multimap s) \mathbin{\&} (\overline{x} \multimap \overline{s})$$

$$\langle\!\langle \top \rangle\!\rangle_s = s \qquad \langle\!\langle \bot \rangle\!\rangle_s = \overline{s}$$

$$\langle\!\langle \neg\, p \rangle\!\rangle_s = \langle\!\langle p \rangle\!\rangle_{s'} \otimes \big((\overline{s'} \multimap s) \mathbin{\&} (s' \multimap \overline{s})\big) \qquad s' \text{ and } \overline{s'} \text{ fresh}$$

$$\left.\begin{aligned} \langle\!\langle p \wedge q \rangle\!\rangle_s &= \langle\!\langle p \rangle\!\rangle_{s_1} \otimes \langle\!\langle q \rangle\!\rangle_{s_2} \otimes \mathtt{conj}(s_1, s_2, s) \\ \langle\!\langle p \vee q \rangle\!\rangle_s &= \langle\!\langle p \rangle\!\rangle_{s_1} \otimes \langle\!\langle q \rangle\!\rangle_{s_2} \otimes \mathtt{disj}(s_1, s_2, s) \end{aligned}\right\} \qquad s_1, \overline{s_1}, s_2, \text{ and } \overline{s_2} \text{ fresh}$$

[1]The full list of these translations can be found in the `tests/prop/qbf` directory of the LI distribution; this directory also includes the program `qbf-nonexp.sml` that can be used to translate any given QBF.

$$\langle\!\langle \forall x.p \rangle\!\rangle_s = \left(!\, x \otimes \langle\!\langle p \rangle\!\rangle_s\right) \oplus \left(!\, \overline{x} \otimes \langle\!\langle p \rangle\!\rangle_s\right)$$
$$\langle\!\langle \exists x.p \rangle\!\rangle_s = \left(!\, x \otimes \langle\!\langle p \rangle\!\rangle_s\right) \,\&\, \left(!\, \overline{x} \otimes \langle\!\langle p \rangle\!\rangle_s\right)$$

The definitions of `conj` *and* `disj` *are as in defn. 7.1. Once again, we define $\langle\!\langle p \rangle\!\rangle$ as $\langle\!\langle p \rangle\!\rangle_s \multimap s$.*

Fact 7.4 (Embedding)**.**
For a closed quantified Boolean formula p,

1. *p is true if and only if $\cdot\,;\langle\!\langle p \rangle\!\rangle_s \Longrightarrow s$; and*
2. *p is false if and only if $\cdot\,;\langle\!\langle p \rangle\!\rangle_s \Longrightarrow \overline{s}$.*

Because this translation allows arbitrary reuse of the variables by means of the exponential operator, the prover is able to use the facilities available to it for handling unrestricted propositions. Therefore, it is able to furnish proofs much faster; even the large example `qbf-3` from before takes mere microseconds. The following table summarises the results of translating the entire collection of formulas in `qbf-1`, `-2` and `-3` from before using the exponential embedding.

	right-biased LIPF				**left-biased LIPF**			
encodings	iters	gen	subs	time	iters	gen	subs	time
`qbf-exp`	1508	1722	140	**0.13**	7948	17610	9590	2.69

In this case the problems are simple enough that the benefits of the left-biased system are still smaller than its overhead over the right-biased system; cumulatively, therefore, the right-biased system performs much better.

7.4.2 Intuitionistic problems

We ran our prover on some problems drawn from the SICS benchmark [104]. These intuitionistic problems were translated into linear logic in two different ways: the first using Girard's original encoding of classical logic in classical linear logic where every subformula is affixed with the exponential (see sec. 2.1.2, and the second using a focus-preserving encoding as described in section 6.4. The former encoding is represented using the suffix `-gir`, and the latter with `-foc`. The problems are ordered in terms of increasing complexity.[2]

We also compared our prover with *Sandstorm*, a focusing inverse method theorem prover for intuitionistic logic implemented by students at Carnegie Mellon University. We might, in principle, have compared to a more traditional external prover such as Vampire[100, 101] or Gandalf [109, 111] here, but such a comparison would be unfair because these external provers, being classical, will not necessarily find intuitionistic proofs.

	right-biased LIFF				**left-biased LIFF**				**SS**
problem	iters	gen	subs	time	iters	gen	subs	time	time
`SICS1-gir`	360	1948	1394	1.312	368	2897	2181	0.6	0.04
`SICS1-foc`	56	365	313	0.056	64	496	415	0.04	
`SICS2-gir`	3035	16391	11732	11.04	3460	27192	20389	5.856	0.06
`SICS2-foc`	489	3133	2688	0.472	616	4672	3902	0.376	
`SICS3-gir`	20958	1131823	810085	762.312	12924	1015552	761517	218.712	1.12
`SICS3-foc`	3377	21659	18646	33.096	2300	17464	14969	23.296	
`SICS4-gir`	×	×	×	×	×	×	×	×	3.89
`SICS4-foc`	8896	57056	49047	87.184	6144	46818	39993	62.24	

The focus-preserving translation is always better than the Girard-translation; however, the complexity of linear logic, particularly the significant complexity of linear contraction, makes it uncompetitive with the intuitionistic prover. These results appear to support a hypothesis that doing purely intuitionistic reasoning in a linear theorem prover is inadvisable. It is a matter of future work to attempt to combine the linear inverse method with the intuitionistic inverse method in a combined procedure. This can either be done at the level of the logic by having separate intuitionistic connectives that operate in the unrestricted context (similar to the approach taken in LNL logic [12]), or the inverse method can be specialised for the image of one of the above translations to use more efficient algorithms that ignore the linear aspects of the sequents. The feasibility of any such process is not supported by any results in this thesis, and should be taken as a statement of conjecture.

[2]It should be noted that this is not the full collection of problems from the SICS benchmark. The particular selection of problems can be seen in the `tests/fo-int` directory of the LI distribution.

7.4.3 Horn clauses from TPTP

For our last set of examples, we selected 20 non-trivial Horn problems from the TPTP version 3.1.1. The selection of problems was not systematic, but we did not constrain our selection to any particular section of the TPTP. The exact list of problems can be found in the `tests/fo-horn` directory of the **li** distribution.

We used the translation described in sec. 6.4.3.

right-biased				**left-biased**			
iters	gen	subs	time	iters	gen	subs	time
4911	314640	287004	**462.859**	6289	704482	526207	638.818

For Horn problems, the right-biased system, which models hyperresolution, performs better than the left-biased system, which models SLD resolution. This observation is not unprecedented— the Gandalf system switches to a Hyperresolution strategy for Horn theories [110]. The likely reason is that in the left-biased system, unlike in SLD resolution system, the derived rule renames the input sequent rather than the rule itself.

Chapter summary *This chapter presents several experiments performed on the provers and algorithms described in earlier chapters.*

Chapter 8

Conclusions

This thesis has the following major contributions.

1. We construct a forward propositional linear sequent calculus with special attention paid to the resource management issues in the forward direction. In particular, we show how to handle structural non-determinism by means of weak linear contexts and weak sequents. The resulting calculus can be used in an inverse method prover.

2. We show how to extend contraction to the first-order case in the presence of additives by means of an algorithmic contraction procedure. The standard lifting procedure is adapted for this new form of contraction to produce a forward calculus of free variables and explicit unification.

3. The sequent calculus is further generalised with a notion of derived inference rules using focused derivations [7]. We first reconstruct focusing from first principles, extending existing intuitionistic focusing calculi with the concept of focusing bias for atomic propositions. We show that altering the focusing bias for atoms gives rise to different derived rules, which correspond closely to forward and backward readings of implications. Finally we show how the focused inverse method simulates hyperresolution and SLD resolution for Horn theories, and show how to translate from intuitionistic to linear logic while preserving focus.

4. Finally, we substantiate the claims of this thesis with an implementation of the various calculi and perform an experimental evaluation on a number of problems

drawn both from applications of linear logic and from proof-theoretic investigations into linear logic. These experiments provide empirical support for the choice of combining the inverse method with focused derivations.

8.1 Future work

Regarding the implementation The implementation of the LI provers has several avenues for further improvement. One aspect that has been entirely neglected is the use of linearity to optimise common operations such as indexing and subsumption. The hierarchical tests (defn. 4.11) are a first attempt at exploiting linearity, but it exploits the information available from multiplicities in only a weak way—only the propositional labels are examined and the term structure is ignored. A better approach might be to represent the contexts themselves as indexed data structures (for example, substitution trees), so one might more accurately detect incompatible multiplicities.

The indexing data structure currently used in the prover is the basic substitution tree structure from first-order classical resolution provers [46, 99]. We conjecture that annotating the data structure with multiplicity information at the internal nodes can improve failure detection. The index can also be improved by utilising features of the term structure to select the indexed proposition; several such proposals for improving the efficiency of subsumption have been made by Tammet [111].

Regarding the focusing calculus As mentioned in sec. 6.4.5, a primary open question regarding our focusing calculus is whether the simulation of hyperresolution extends to a fuller fragment such as classical linear logic. We are optimistic that such an extension can be found.

The space of focusing-aware translations, and the resulting behaviour of derived inference rules, has only begun to be explored. Several sequent calculi for intuitionistic logics such as LJK and LJT variously contain or lack features of focusing, which can be explained by means of selective affixion of the exponential ! in translations to intuitionistic linear logic [67]. It would be interesting to attempt similar translations from LO to the full linear logic that preserves semantics such as bottom-up evaluation à la Bozzano and

Delzanno [15, 18].

Extensions to the logic One important extension to the linear logic that has been found to be important in practice is support for constraint domains. For practical reasons it is infeasible and wasteful to treat the constraint domains internally in the sequent calculus; instead, one would like to use the growing library of efficient decision procedures and constraint solvers. There are at least two such proposals for extensions to the basic intuitionistic linear logic: ILC [59], which has been proposed for reasoning about pointer programs, and CILL [105], which has been proposed for hybrid robotic planning. The key idea in each case is to extend the standard sequent $\Gamma ; \Delta \Longrightarrow C$ with an additional guarded constraint Ψ, such that the sequent $\Psi \mid \Gamma ; \Delta \Longrightarrow C$ is meaningful only if the guard Ψ is a satisfiable ($\vdash \Psi$).[1] It is fairly simple to construct the forward version of the CILL calculus, but interfacing it with constraint solvers is a significant engineering task. The primary issue is that forward sequents, being local objects, have independent persistent constraint objects, whereas most constraint solvers are designed to deal with a single global mutable constraint object. It is an open question whether existing constraint solvers can be adapted to handle this mismatch.

[1]More precisely, every satisfying assignment to the constraint gives an instance of the linear sequent that should be true.

Bibliography

[1] BLAST. Home page: `http://www-cad.eecs.berkeley.edu/~rupak/blast`.

[2] ELAN. Home page: `http://www.loria.fr/equipes/protheo/SOFTWARES/ELAN/`.

[3] Lygon. Home page: `http://www.cs.rmit.edu.au/lygon/`.

[4] The Maude system. Home page: `http://maude.cs.uiuc.edu/`.

[5] Samson Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111(1&2):3–57, 1993.

[6] Samson Abramsky and Radha Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 15(2):543–574, 1994.

[7] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

[8] Jean-Marc Andreoli. Focussing and proof construction. *Annals of Pure and Applied Logic*, 107:131–163, 2001.

[9] Jean-Marc Andreoli and R. Pareschi. Linear objects: logical processes with built-in inheritance. *New Generation Computing*, 9(3-4):445–473, 1991.

[10] Thomas Ball and Sriram K. Rajamani. Bebop: A symbolic model checker for boolean programs. In *SPIN*, pages 113–130. Microsoft Corporation, 2000.

[11] Andrew Barber and Gordon Plotkin. Dual intuitionistic linear logic. Technical Report ECS-LFCS-96-347, University of Edinburgh, 1996.

[12] Nick Benton. A mixed linear and non-linear logic: Proofs, terms and models. Technical Report 352, University of Cambridge Computer Laboratory, 1994. [65 page version].

[13] Nick Benton, Gavin Bierman, Valeria de Paiva, and Martin Hyland. A term calculus for intuitionistic linear logic. In M. Bezem and G. F. Groote, editors, *Proceedings of the International Conference on Typed Lambda Calculi and Applications (TLCA)*, pages 75–90, Utrecht, The Netherlands, March 1993. Springer-Verlag LNCS 664.

[14] Marco Bozzano. *A Logic-Based Approach to Model Checking of Parameterized and Infinite-State Systems*. PhD thesis, DISI, Università di Genova, 2002.

[15] Marco Bozzano. *A Logic-Based Approach to Model Checking of Parameterized and Infinite-State Systems*. PhD thesis, DISI, Università di Genova, 2002.

[16] Marco Bozzano and Giorgio Delzanno. Algorithmic verification of invalidation-based protocols. In *Proceedings of CAV*, Copenhagen, Denmark, July 2002.

[17] Marco Bozzano and Giorgio Delzanno. Automated protocol verification in linear logic. In *Proceedings of PPDP*, Pittsburgh, Pennsylvania, October 2002.

[18] Marco Bozzano, Giorgio Delzanno, and Maurizio Martelli. Model checking linear logic specifications. *TPLP*, 4(4–6):573–619, 2004.

[19] Stephen D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, July 1984.

[20] Iliano Cervesato. Petri nets and linear logic: a case study for logic programming. In M. Apuente and M.I. Sessa, editors, *Proceedings of the Joint Converence on Declarative Programming (GULP-PRODE)*, pages 313–318, Marina di Vietri, Italy, September 1995. Palladio Press.

[21] Iliano Cervesato. A specification language for crypto-protocols based on multi-set rewriting, dependent types and subsorting. In G. Delzanno, S. Etalle, and M. Gabbrielli, editors, *Workshop on Specification, Analysis and Validation for Emerging Technologies — SAVE'01*, pages 1–22, Paphos, Cyprus, December 2001.

[22] Iliano Cervesato, Joshua S. Hodas, and Frank Pfenning. Efficient resource management for linear logic proof search. In R. Dyckhoff, H. Herre, and P. Schroeder-Heister, editors, *Proceedings of the 5th International Workshop on Extensions of Logic Programming*, pages 67–81, Leipzig, Germany, March 1996. Springer-Verlag LNAI 1050.

[23] Iliano Cervesato, Joshua S. Hodas, and Frank Pfenning. Efficient resource management for linear logic proof search. *Theoretical Computer Science, special issue on Proof Search in Type-Theoretic Languages*, 232(1-2):133–163, February 2000.

[24] Iliano Cervesato and Frank Pfenning. A linear logical framework. In E. Clarke, editor, *11th Annual Symposium on Logic in Computer Science — LICS'96*, pages 264–275, New Brunswick, NJ, 27–30 July 1996. IEEE Computer Society Press. This work appeared as Preprint 1834 of the Department of Mathematics of Technical University of Darmstadt, Germany.

[25] Iliano Cervesato, Frank Pfenning, David Walker, and Kevin Watkins. A concurrent logical framework II: Examples and applications. Technical Report CMU-CS-02-102, Carnegie Mellon University, March 2002. At: `http://www-2.cs.cmu.edu/~fp/papers/CMU-CS-02-102.pdf`.

[26] Sagar Chaki, Sriram K. Rajamani, and Jakob Rehof. Types as models: Model checking message-passing programs. In *Proceedings of POPL 2002*, January 16–18 2002.

[27] Bor-Yuh Evan Chang, Kaustuv Chaudhuri, and Frank Pfenning. A judgmental analysis of linear logic. Technical Report CMU-CS-03-131-R, Carnegie Mellon University, 2003, revised 2004.

[28] Kaustuv Chaudhuri and Frank Pfenning. A focusing inverse method theorem prover for first order linear logic. In *Proceedings of CADE-20*, pages 69–83, Tallinn, Estonia, July 2005. Springer-Verlag LNCS-3632.

[29] J. Christian. Fast Knuth-Bendix completion: A summary. In N. Dershowitz, editor, *Proceedings of the 3rd International Conference on Rewriting Techniques and Applications (RTA)*, pages 551–555. Springer-Verlag LNCS 355, 1986.

[30] J. Christian. Flatterms, discrimination nets, and fast term rewriting. *Journal of Automated Reasoning*, 10(1):95–113, 1993.

[31] Edward M. Clarke and O. Grumberg. Avoiding the state explosion problem in temporal logic model checking algorithms. In *Proceedings of PODC'87*, pages 294–303. ACM Press, 1987.

[32] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. Lkq and lkt: sequent calculi for second order logic based upon dual linear decompositions of classical implication. In Jean-Yves Girard and Yves Lafont, editors, *Proceedings of the Workshop on Linear Logic*, pages 211–224. London Mathematical Society Lecture Notes 222, Cambridge University Press, 1995.

[33] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. A new deconstructive logic: Linear logic. *Journal of Symbolic Logic*, 62(3):755–807, 1997.

[34] Satyaki Das, David L Dill, and Seungjoo Park. Experience with predicate abstraction. *Computer Aided Verification*, pages 160–171, 1999.

[35] Anatoli Degtyarev and Andrei Voronkov. The inverse method. In James Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, chapter 4, pages 179–272. MIT Press, September 2000.

[36] Kevin Donnelly, Tyler Gibson, Neel Krishnaswami, Stephen Magill, and Sungwoo Park. The inverse method for the logic of bunched implications. In F. Baader and Andrei Voronkov, editors, *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, pages 466–480, Montevideo, Uruguay, March 2005. Springer LNCS 3452.

[37] Roy Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *Journal of Symbolic Logic*, 57:795–807, 1992.

[38] Harvey Friedman. Classically and intuitionistically provably recursive functions. In D.S. Scott and G.H. Muller, editors, *Higher Set Theory*, pages 21–27. Springer-Verlag LNM 699, 1978.

[39] Michael R. Garey and David S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman, San Francisco, 1979. ISBN 0-7167-1045-5.

[40] Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935. English translation in M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131, North-Holland, 1969.

[41] Gerhard Gentzen. *Collected papers of Gerhard Gentzen*. North-Holland, Amsterdam, 1968.

[42] Jean-Yves. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[43] Jean-Yves Girard. On the unity of logic. *Annals of Pure and Applied Logic*, 59:201–217, 1993.

[44] Jean-Yves Girard. Locus solum: from the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11:301–506, 2001.

[45] G. Godelfroid. Using partial orders to improve automatic verification methods. In *Proceedings of CAV*, 1990.

[46] Peter Graf. *Term Indexing*. Springer LNAI 1053, 1996.

[47] Susanne Graf and Hassen Saidi. Construction of abstract state graphs with pvs. In O. Grumberg, editor, *Prceedsings of CAV*, volume 1254, pages 72–83. Springer-Verlag, 1997.

[48] James Harland and David Pym. The uniform proof-theoretic foundations of linear logic programming. In V. Saraswat and K. Ueda, editors, *Proceedings of the International Logic Programming Symposium*, pages 034–318, San Diego, California, October 1991.

[49] James Harland and David J. Pym. Resource-distribution via boolean constraints. In W. McCune, editor, *Proceedings of CADE-14*, pages 222–236, Townsville, Australia, July 1997. Springer-Verlag LNAI 1249.

[50] James Harland and Philip Winikoff. Deterministic resource management for the linear logic programming Lygon. Technical Report TR 94/23, Melbourne University, Department of Computer Science, 1994.

[51] Robert Harper and Frank Pfenning. On equivalence and canonical forms in the lf type theory. Technical Report CMU-CS-00-148, Carnegie Mellon University, Pittsburgh, PA, 2000.

[52] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 1978.

[53] Joshua S. Hodas. Lolli: an extension of λProlog with linear logic context management. In Dale Miller, editor, *Proceedings of the 1992 workshop on the λProlog programming language*, Philadelphia, 1992.

[54] Joshua S. Hodas. Logic programming with multiple context management schemes. In Roy Dyckhoff, editor, *Fourth International Workshop on Extensions of Logic Programming*, pages 171–182, St. Andrews, United Kingdom, 1993. Springer-Verlag LNCS 360.

[55] Joshua S. Hodas. *Logic Programming in Intuitionistic LInear Logic: Theory, Design, and Implementation*. PhD thesis, University of Pennsylvania, 1994.

[56] Joshua S. Hodas and Dale Miller. Logic programming in a fragment of linear logic. *Journal of Information and Computation*, 110(2):327–365, 1994.

[57] Jakob M. Howe. *Proof search issues in some non-classical logics*. PhD thesis, University of St. Andrews, September 1999.

[58] Radhakrishna Jagadeesan, Gopalan Nadathur, and Vijay Saraswat. Testing concurrent sytems: An interpretation of intuitionistic logic. In *Proceedings of the 25th Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 517–528. Springer-Verlag LNCS 3821, 2005.

[59] Limin Jia and David Walker. ILC: A foundation for automated reasoning about pointer programs. In P. Sestoft, editor, *Proceedings of the 15th European Symposium on Programming Languages and Systems (ESOP'06)*, pages 131–145. Springer-Verlag LNCS 3924, March 2006.

[60] John Arnold Kalman. *Automated reasoning with Otter*. Rinton Press, Princeton, NJ, 2001. With a foreword by Larry Wos.

[61] R. Kowalski and D. Kuehner. Linear resolution with selection function. *Artificial Intelligence*, 2:227–260, 1971.

[62] Yves Lafont and Thomas Streicher. Games semantics for linear logic. In *Proceedings of the Sixth Annual IEEE Symposium on Logic in Computer Science*, Amsterdam, The Netherlands, July 1991. IEEE Computer Society Press, Los Amitos, California.

[63] James R. Larus, Sriram K. Rajamani, and Jakob Rehof. Behavioral types for structured asynchronous programming. Technical report, Microsoft Research, November 2001. Contact authors for access.

[64] Olivier Laurent. *Etude de la polarisation en logique*. PhD thesis, University of Aix-Marseille, March 2002.

[65] Olivier Laurent. Classical isomorphisms of types. *Mathematical Structures in Computer Science*, 15(5):969–1004, 2005.

[66] Paul B. Levy. Jumbo lambda-calculus. In V. Sassone M. Bugliesi, B. Preneel and I. Wegener, editors, *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2, pages 444–455, Venice, Italy, July 2006. Springer-Verlag LNCS 4052.

[67] Chuck Liang and Dale Miller. On focusing and polarities in linear logic and intuitionistic logic, December 2006. Draft manuscript.

[68] Patrick D. Lincoln, John C. Mitchell, Andre Scedrov, and Natarajan Shankar. Decision problems for propositional linear logic. *Annals of Pure and Applied Logic*, 56:239–311, 1992.

[69] K. L.McMillan. A technique of state space search based on unfolding. *Formal Methods in System Design: An International Journal*, 1995.

[70] Heiko Mantel and Jens Otten. LinTAP: A tableau prover for linear logic. In A. Murray, editor, *International Conference TABLEAUX'99*, pages 217–231, New York, June 1999. Springer-Verlag LNAI 1617.

[71] Narciso Martí-Oliet and José Meseguer. Action and change in rewriting logic. In R. Pareschi and B. Fronhöfer, editors, *Dynamic Worlds*, pages 1–53. Kluwer, Dordrecht, 1999.

[72] Per Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1):11–60, 1996. Lecture notes to a short course at Università degli Studi di Siena, April 1983.

[73] S. Maslov. The inverse method of establishing deducibility in the classical predicate calculus. *Soviet Mathematical Doklady*, 5:1420–1424, 1964.

[74] W. McCune. Experiments with discrimination-tree indexing and path indexing for term retrieval. *Journal of Automated Reasoning*, 9(2):147–167, 1992.

[75] Daniel Méry. *Preuves et Sémantiques dans des Logiques de Ressources*. PhD thesis, Université Henri Poincaré, Nancy, France, November 2004.

[76] Stephan Merz. On the logic of TLA+. *Special Issue of Computers and Informatics on the semantics of specification formalisms*, 2003.

[77] Dale Miller. The π-calculus as a theory in linear logic: Preliminary results. In E. Lamma and P. Mello, editors, *Proceedings on the Workshop on Extensions of Logic Programming*, pages 242–265. Springer Verlag LNCS 660, 1992.

[78] Dale Miller. Forum: A multiple-conclusion specification logic. *Theoretical Computer Science*, 165(1):201–232, 1996.

[79] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[80] Dale A. Miller, Gopalan Nadathur, and Andre Scedrov. Hereditary Harrop formulas and uniform proof systems. In David Gries, editor, *Proceedings of the 2nd IEEE Symposium on Logic in Computer Science*, pages 98–105, Ithaca, New York, 1987.

[81] Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[82] Robin Milner. *Communicating and Mobile Systems: the π-calculus*. Cambridge University Press, 1999.

[83] Grigori Mints. Resolution calculus for the first order linear logic. *Journal of Logic, Language and Information*, 2(1):59–83, 1993.

[84] Grigori Mints. Resolution strategies for the intuitionistic logic. In *Constraint Programming*, NATO ASI Series F, pages 289–311. Springer-Verlag, 1994.

[85] Kedar S. Namjoshi. *Ameliorating the state-space explosion problem*. PhD thesis, University of Texas at Austin, 1998.

[86] Sara Negri. A normalizing system of natural deduction for intuitionistic linear logic. *Mathematical Logic*, 41:789–810, September 2000.

[87] Sara Negri. Varieties of linear calculi. *Journal of Philosophical Logic*, 31:569–590, 2002.

[88] Sara Negri and Roy Dykhoff. Admissibility of structural rules for contraction-free systems of intuitionistic logic. *Journal of Symbolic Logic*, 65:1499–1518, December 2000.

[89] Peter O'Hearn. On bunched typing. *Journal of Functional Programming*, 13(4):747–796, 2003.

[90] D. Pelled. All from one, one for all: on model checking using representatives. In *Proceedings of CAV*, 1998.

[91] C. A. Petri. Fundamentals of a theory of asynchronous information flow. In *Proceedings of the IFTP Congress 62*, 1963.

[92] Frank Pfenning. Structural cut elimination in linear logic. Technical Report CMU-CS-94-222, Carnegie Mellon University, December 1994.

[93] Frank Pfenning. Structural cut elimination in linear logic. Technical Report CMU-CS-94-222, Carnegie Mellon University, December 1994.

[94] Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11(4):511–540, 2001.

[95] Jeff Polakow. *Ordered Linear Logic and Applications*. PhD thesis, Carnegie Mellon University, August 2001.

[96] Jeff Polakow and Frank Pfenning. Relating natural deduction and sequent calculus for intuitionistic non-commutative linear logic. In Andre Scedrov and Achim Jung, editors, *Proceedings of the 15th Conference on Mathematical Foundations of Programming*

Semantics, New Orleans, Louisiana, April 1999. Electronic Notes in Theoretical Computer Science, Volume 20.

[97] Dag Prawitz. Ideas and results in proof theory. In Jens Erik Fenstad, editor, *Proceedings of the 2nd Scandinavian Logic Symposiym*, pages 235–307, Amsterdam, June 1970. North Holland.

[98] David J. Pym and James Harland. A uniform proof-theoretic investigation of linear logic. *Journal of Logic and Computation*, 4(2):175–207, April 1994.

[99] I. V. Ramakrishnan, R. C. Sekar, and Andrei Voronkov. Term indexing. In *Handbook of Automated Reasoning*, pages 1853–1964. Elsevier and MIT Press, 2001.

[100] Alexander Riazanov and Andrei Voronkov. Vampire. In *Proceedings of CADE-16*, pages 282–286. Spring-Verlag LNAI 1632, 1999.

[101] Alexander Riazanov and Andrei Voronkov. Vampire 1.1 (system description). In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR)*, pages 376–380. Springer-Verlag LNAI 2083, 2001.

[102] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[103] James Alan Robinson. Automatic deduction with hyper-resolution. *International Journal of Computational Mathematics*, 1:227–234, 1965.

[104] Dan Sahlin, Torkel Franzén, and Seif Haridi. An intuitionistic predicate logic theorem prover. *Journal of Logic and Computation*, 2(5):619–656, 1992.

[105] Uluç Saranli and Frank Pfenning. Using constrained intuitionistic linear logic for hybrid robotic planning problems, September 2006. Draft manuscript.

[106] Harold Schellinx. *The Noble Art of Linear Decorating*. PhD thesis, University of Amsterdam, February 1994.

[107] Danny D. Sleator and Robert E. Tarjan. Self-adjusting binary search trees. *Journal of the ACM*, 32(3):652–686, 1985.

[108] Tanel Tammet. A resolution theorem prover for intuitionistic logic. In M. McRobbie and J. Slaney, editors, *Proceedings of CADE-13*, pages 2–16, New Brunswick, New Jersey, 1996. Springer-Verlag LNCS 1104.

[109] Tanel Tammet. Gandalf. *Journal of Automated Reasoning*, 18(2):199–204, 1997.

[110] Tanel Tammet. Resolution, inverse method and the sequent calculus. In *Proceedings of the 5th Kurt Gödel Colloquim on Computational Logic and Proof Theory (KGC'97)*, pages 65–83, Vienna, Austria, 1997. Springer-Verlag LNCS 1289.

[111] Tanel Tammet. Towards efficient subsumption. In *Proceedings of CADE-15*, pages 427–441, 1998.

[112] Tannel Tammet. Proof strategies in linear logic. *Journal of Automated Reasoning*, 12(3):273–304, 1994.

[113] Naoyuki Tamura. Llprover. At: `http://bach.istc.kobe-u.ac.jp/llprover`.

[114] A. Valmari. A stubborn attack on state explosion. In *Proceedings of CAV*, 1990.

[115] M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.

[116] Andrei Voronkov. Theorem proving in non-standard logics based on the inverse method. In D. Kapur, editor, *Proceedings of the CADE-11*, pages 648–662, Saratoga Springs, New York, 1992. Springer-Verlag LNCS 607.

[117] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A concurrent logical framework I: Judgments and properties. Technical Report CMU-CS-02-101, Carnegie Mellon University, March 2002.

[118] Michael Winikoff and James Harland. Implementing the linear logic programming language Lygon. In *Proceedings of the International Logic Programming Symposium (ILPS)*, pages 66–80, December 1995.

[119] Noam Zeilberger. On the unity of duality. Unpublished manuscript, 2006. Available at: `http://www.cs.cmu.edu/~noam/research/unity-duality.pdf`.